

(19) 日本国特許庁 (J P)

(12) 公表特許公報 (A)

(11) 特許出願公表番号

特表平8-502865

(43) 公表日 平成8年(1996)3月26日

(51) Int.Cl. ⁶	識別記号	庁内整理番号	F I
H 0 4 N 7/24			
H 0 3 M 7/30	B	9382-5K	
H 0 4 B 14/04	Z	8221-5J	
H 0 4 N 1/41	B	9070-5C	
		4228-5C	
		H 0 4 N 7/13	Z
	審査請求	未請求	予備審査請求 有 (全 72 頁) 最終頁に続く

(21) 出願番号	特願平6-507421
(86) (22) 出願日	平成5年(1993)8月31日
(85) 翻訳文提出日	平成7年(1995)3月1日
(86) 国際出願番号	PCT/US 93/08235
(87) 国際公開番号	WO 94/06099
(87) 国際公開日	平成6年(1994)3月17日
(31) 優先権主張番号	07/938, 959
(32) 優先日	1992年9月1日
(33) 優先権主張国	米国 (US)

(71) 出願人	アプル・コンピュータ・インコーポレーテッド アメリカ合衆国 95014 カリフォルニア 州・カップチャーノ・マリアニイ アヴェニ ユ・20225
(72) 発明者	ノーマイル, ジェイムズ・オリバー アメリカ合衆国 94087 カリフォルニア 州・サニー ヴェイル・テナカ プレイ ス・ナンバープイ 1・1575
(74) 代理人	弁理士 山川 政樹 (外5名)

最終頁に続く

(54) 【発明の名称】 改良されたベクトルの量子化

(57) 【要約】

データの圧縮に関するコードブックを構築するベクトル量子化 (VQ) のための改良された方法と装置。コードブック 600 又は「ツリー」は、N 個の初期ノード (610) を設定し、残りのコードブックをバイナリ・コードブック (650) として生成することによって初期化される。子供の項目 (670) が、最大歪、個体数等の様々な属性の決定に基づいて分割される。データから得られたベクトルは子供ノードと関連づけられ、次に代表的子供の項目が再計算される。この分割/再関連は、前の子供と現在の子供に関する誤差の差が閾値より小さくなるまでくり返し続けられる。この分割と再関連処理は、最大数のターミナル・ノードがツリー内に作成されたり、合計誤差又は歪が閾値に達する等の基準まで続けられる。データは次にコードブックとコードブックを参照するインデックスを含む圧縮されたビット・ストリームとして伝送される。

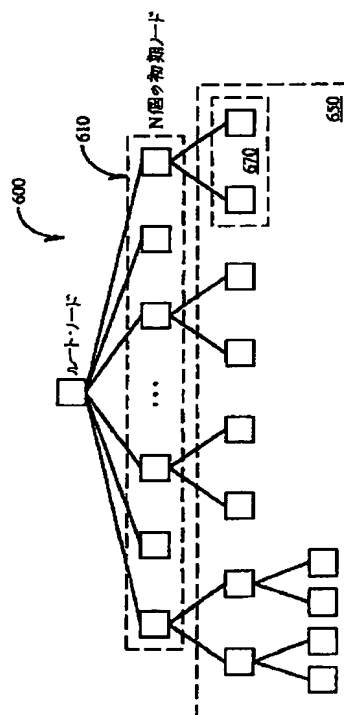


FIG. 6

BEST AVAILABLE COPY

【特許請求の範囲】

1. a. ベクトル量子化ツリーのN個の初期ノードを初期化するステップと、
、
b. イメージからベクトルをサンプリングするステップと、
c. 上記イメージからサンプリングされた最良の代表サンプルである上記ベクトル量子化ツリーのノードを決定するステップと、
d. 上記ベクトルを上記ベクトル量子化ツリーの上記ノードに関連づけるステップと、
e. 上記イメージから次のベクトルをサンプリングするステップと、
f. 上記イメージからサンプリングするベクトルがなくなるまでc-fをくり返し、上記次のベクトルが上記ベクトルになるステップと、
g. 上記ツリーのどのノードが上記ツリー内で最も歪んでいるかを決定するステップと、
h. 上記最も歪んだノードを2つの子供ノードに分割するステップと、
i. 上記最も歪んだノードと関連するベクトルの第1の部分を上記子供ノードの第1と関連づけ、上記最も歪んだノードと関連するベクトルの第2の部分を上記子供ノードの第2と関連づけるステップと、
j. 上記ベクトルの第1及び第2の部分と比較した2つの子供ノードの現在の誤差を決定するステップと、
k. 現在の誤差と前の誤差の間の誤差の変化が誤差の閾値より小さいとき、ステップ1に進み、その他の場合は上記第1及び第2の子供の新しい値を決定し、ステップiに進み、現在の誤差が前の誤差になるステップと、
l. 上記ベクトル量子化ツリー内のターミナル・ノードの数が所望の個体数に達するまでステップg-1をくり返すステップと、
m. インデックスを上記ベクトル量子化ツリーのターミナル・ノードのそれぞれに関連づけるステップと、
から成るイメージのベクトル量子化方法。
2. 上記入力イメージの上記サンプリングされたベクトルのそれぞれと関連する上記ベクトル量子化ツリーのターミナル・ノードのインデックスを表す上記ベ

クトル量子化ツリーからの一連のインデックスを伝送するステップを更に含む請

求項1に記載のイメージのベクトル量子化方法。

3. 最も歪んだノードを決定するステップが、そのノードと関連した上記入力イメージからサンプリングされたベクトルのそれぞれと比較されたノードの平均歪が、閾値を超えたかどうか判定するステップを含む請求項1に記載のイメージのベクトル量子化方法。

4. 最も歪んだノードを決定するステップが、そのノードと関連した上記入力イメージからサンプリングされたベクトルのそれぞれと比較されたノードの合計歪が、閾値を超えたかどうか判定するステップを含む請求項1に記載のイメージのベクトル量子化方法。

5. 最も歪んだノードを決定するステップが、上記ノードと関連した入力イメージからのサンプリングされた個体数が閾値を超えたかどうか判定するステップを含む請求項1に記載のイメージのベクトル量子化方法。

6. 最も歪んだノードを決定するステップが、そのノードと関連した上記入力イメージからサンプリングされたベクトルのそれぞれと比較されたノードのパーセンテージ歪が、閾値を超えたかどうか判定するステップを含む請求項1に記載のイメージのベクトル量子化方法。

7. 最も歪んだノードを決定するステップが、そのノードと関連した上記入力イメージからサンプリングされたベクトルのそれぞれと比較されたノードの最大歪が、閾値を超えたかどうか判定するステップを含む請求項1に記載のイメージのベクトル量子化方法。

8. 最も歪んだノードを決定するステップが、そのノードと関連した上記入力イメージからサンプリングされたベクトルのそれぞれと比較されたノードの最大歪の最小歪に対する比が、閾値を超えたかどうか判定するステップを含む請求項1に記載のイメージのベクトル量子化方法。

9. 上記イメージからサンプリングされたベクトルの最良の代表サンプルであるノード決定するステップがサンプリングされたベクトルとそのノードの間の平均2乗誤差を決定することを含む請求項1に記載のイメージのベクトル量子化方

法。

10. 平均2乗誤差を決定するステップが、上記ベクトル量子化の初期の部分

では大きな誤差ほど重く加重され、上記ベクトル量子化の後の部分では大きな誤差ほど軽く加重される請求項9に記載のイメージのベクトル量子化方法。

11. N個の初期ノードを作成するステップが前のイメージについて行われた前のベクトル量子化からのN個の初期ノードを用いることを含む請求項1に記載のイメージのベクトル量子化方法。

12. 上記最も歪んだノードを分割する前に、上記入力イメージからのベクトル・サンプルに対して疑似ランダムに生成された値を加えるステップを更に含む請求項1に記載のイメージのベクトル量子化方法。

13. 上記イメージからサンプリングされたベクトルの最良の代表サンプルであるノードを決定するステップが、そのイメージからサンプリングされたベクトルに最も近い輝度とクロミナンス (YUV) 値を有するノードを決定することを含む請求項1に記載のイメージのベクトル量子化方法。

14. 上記イメージの異なったゾーンに対して別々のベクトル量子化ツリーを生成するステップを更に含む請求項1に記載のイメージのベクトル量子化方法。

15. 可変サイズを有する上記イメージの異なったゾーンを決定するステップを更に含む請求項14に記載のイメージのベクトル量子化方法。

16. 一連のイメージに対して適用され、上記一連のイメージ内で場面変化が検出されたとき、新しいベクトル量子化ツリーの作成が行われる請求項1に記載のイメージのベクトル量子化方法。

17. a. ベクトル量子化ツリーのN個の初期ノードを初期化する手段と、

b. イメージからベクトルをサンプリングする手段と、

c. 上記イメージからサンプリングされた最良の代表サンプルである上記ベクトル量子化・ツリーのノードを決定する手段と、

d. 上記ベクトルを上記ベクトル量子化ツリーの上記ノードに関連づける手段と、

e. 上記イメージから次のベクトルをサンプリングする手段と、

- f. 構成要素 c-e を上記イメージからサンプリングされるべきベクトルがなくなるまで活性化し、上記次のベクトルが上記ベクトルとなる手段と、
- g. 上記ツリーのどのノードが上記ツリー内で最も歪んでいるかを決定する手段と、

段と、

- h. 上記最も歪んだノードを2つの子供ノードに分割する手段と、
 - i. 上記最も歪んだノードと関連するベクトルの第1の部分を上記子供ノードの第1と関連づけ、上記最も歪んだノードと関連するベクトルの第2の部分を上記子供ノードの第2と関連づける手段と、
 - j. 上記ベクトルの第1及び第2の部分と関連した2つの子供ノードの現在の誤差を決定する手段と、
 - k. 上記第1及び上記第2の子供ノードの新しい値を決定し、継続的に構成要素 i-j を活性化し、もし現在の誤差と前の誤差の間の誤差の変化が誤差の閾値より大きければ現在の誤差が上記前の誤差となる手段と、
 - l. 上記ベクトル量子化ツリーのターミナル・ノードの数が所望の個体数に達するまで継続的に構成要素 g-1 を活性化する手段と、
 - m. インデックスを上記ベクトル量子化ツリーのターミナル・ノードのそれぞれに関連づける手段と、
- を備えたイメージのベクトル量子化装置。

- 18. a. ベクトル量子化ツリーのN個の初期ノードを初期化する手段と、
- b. イメージからベクトルをサンプリングする手段と、
- c. 上記イメージからサンプリングされた最良の代表サンプルである上記ベクトル量子化ツリーのノードを決定する手段と、
- d. 上記ベクトルを上記ベクトル量子化ツリーの上記ノードに関連づける手段と、
- e. 上記ツリーのターミナル・ノードの数が所望の個体数に達するまでくり返し、上記ツリー内の最悪のノードを決定することによってベクトル量子化ツリー内に新しいノードを作成し、上記ノードを分割し、上記ベクトルを上記ベクトル量子化ツリーの上記ノードと再度関連づける手段と、

を備えたイメージのベクトル量子化装置。

19. a. ベクトル量子化ツリーのN個の初期ノードを初期化する手段と、
- b. イメージからベクトルをサンプリングする手段と、
- c. 上記イメージからサンプリングされた最良の代表サンプルである上記ベク

トル量子化ツリーのノードを決定する手段と、

- d. 上記ベクトルを上記ベクトル量子化ツリーの上記ノードに関連づける手段と、
- e. 上記ツリーが上記ベクトルと比較して所望の歪に達するまでくり返し、上記ツリー内の最悪のノードを決定することによってベクトル量子化ツリー内に新しいノードを作成し、上記ノードを分割し、上記ベクトルを上記ベクトル量子化ツリーの上記ノードと再度関連づける手段と、

を備えたイメージのベクトル量子化装置。

20. a. ベクトル量子化ツリーのN個の初期ノードを初期化する手段と、
- b. イメージからベクトルをサンプリングする手段と、
- c. 上記イメージからサンプリングされた最良の代表サンプルである上記ベクトル量子化ツリーのノードを決定する手段と、

- d. 上記ベクトルを上記ベクトル量子化ツリーの上記ノードに関連づける手段と、

- e. 上記ツリーのターミナル・ノードの数が所望の個体数に達するまでくり返し、上記ツリー内の最悪のノードを決定することによってベクトル量子化ツリー内に新しいノードを作成し、上記ノードを2つ以上の子供ノードに分割し、上記ベクトルを上記ベクトル量子化ツリー内の上記子供ノードと再度関連づける手段と、

を備えたイメージのベクトル量子化装置。

21. a. イメージの異なった領域が閾値に基づいてコード化されるべきかどうかを決定し、上記閾値は、上記領域内のブロックがコード化時に類似のタイプであるかどうかを含む手段と、

- b. 上記イメージの上記別々の領域を別々にコード化する手段と、

c. 上記イメージの上記別々の領域が別々にコード化されたことを示し、上記別々の領域のそれぞれに対する別々のコードブックを参照する手段を含み、更に上記別々の領域の位置を示す手段と、

を備えたイメージをコード化する装置。

22. 上記類似のタイプは、上記領域のブロックが上記イメージの変化ブロッ

クを含むかどうかを含む請求項21に記載のイメージをコード化する装置。

23. 上記類似のタイプは、上記領域のブロックが上記イメージの無変化ブロックを含むかどうかを含む請求項21に記載のイメージをコード化する装置。

24. 上記イメージの上記無変化ブロックは上記イメージ内に空間的無変化ブロックを含む請求項23に記載のイメージをコード化する装置。

25. 上記イメージの上記無変化ブロックは、上記イメージと比較された前のイメージからの時間的無変化ブロックを含む請求項23に記載のイメージをコード化する装置。

26. 上記類似のタイプは、上記別々の領域の1つが見かけ上空間的にスムーズであり、上記別々の領域の第2が見かけ上空間的にディテイルであるかどうかを含む請求項21に記載のイメージをコード化する装置。

27. 上記コード化の手段がベクトル量子化手段を含む請求項26に記載のイメージをコード化する装置。

28. 上記別々の領域が、複数のイメージにおいて別々にコード化され、上記領域に対する別々のコードブックが上記複数のイメージで共用される請求項21に記載のイメージをコード化する装置。

29. a. イメージの異なった領域が、閾値に基づいてコード化されるべきかどうかを決定し、上記閾値は、上記領域内のブロックがコード化時に類似のタイプであるかどうかを含む手段と、

b. 上記イメージの上記別々の領域を別々にコード化する手段と、

c. 上記イメージの上記別々の領域が別々にコード化されたことを示し、上記別々の領域のそれぞれに対するコードブックを参照する手段を含み、更に上記別々の領域の位置を示す手段と、

を備えたイメージをコード化する装置。

30. 上記コードブックは上記複数の別々の領域に対して共用される請求項29に記載のイメージをコード化する装置。

31. 上記別々の領域のいくらかは、上記別々の領域のいくらかをコード化する誤差が上記閾値を超えると下位領域に更に分割され、上記下位領域のそれぞれの別々のコード化となる請求項29に記載のイメージをコード化する装置。

32. 上記誤差が平均2乗誤差を含む請求項31に記載のイメージをコード化する装置。

33. もし上記別々の領域をコード化する誤差が閾値を超えれば、上記別々の領域は帰納的に他の別々の領域に分割される請求項29に記載のイメージをコード化する装置。

34. 上記別々の領域のそれぞれに対するコードブックはサイズが異なる請求項29に記載のイメージをコード化する装置。

35. a. 前のイメージからの前のコードブックを用いて現在のイメージを第1のコード化されたイメージとしてコード化する手段と、

b. 上記前のコードブックを用いる第1のコード化されたイメージが上記現在のイメージの良好な近似であるかどうか判定する手段と、

c. 上記前のコードブックを更新して上記現在のイメージに対する現在のコードブックを作成し、上記決定する手段に応答して上記第1のコード化されたイメージが上記現在の良好な近似でないことを判定する手段と、

d. 上記現在のイメージを上記現在のコードブックを用いて第2のコード化されたイメージとしてコード化し、上記決定する手段に応答して上記第1のコード化されたイメージが上記現在のイメージの良好な近似でないことを判定する手段と、

を備えた現在のイメージをコード化する装置。

36. 上記第1のコード化されたイメージが上記現在のイメージの良好な近似でないことを判定する手段が、上記第1のコード化されたイメージから生成されたコード化された第1のイメージと上記現在のイメージとの間の平均2乗誤差が

閾値より大きいかどうか判定する手段を含む請求項35に記載の現在のイメージをコード化する装置。

37. 上記閾値が、前のコード化されたイメージから生成された第2のコード化されたイメージと前のコード化されていないイメージとの間の平均2乗誤差を含む請求項36に記載の現在のイメージをコード化する装置。

38. 上記閾値が現在のフレームの前のN個のコード化されたフレームと関連するN個のコード化されないフレームの平均2乗誤差の平均値を含む請求項36

に記載の現在のイメージをコード化する装置。

39. 上記閾値が、上記現在のイメージからのベクトルと比較して大きな平均2乗誤差を有する上記第1のコード化されたイメージから生成されたデコードされたベクトル数を含む請求項36に記載の現在のイメージをコード化する装置。

40. もし上記第1のコード化されたイメージから生成されたデコードされたベクトルが上記現在のイメージと比較して大きな平均2乗誤差を有するならば、上記更新する手段が、上記前のコードブックの特定項目を更新して上記現在のコードブックを生成する手段を含む請求項36に記載の現在のイメージをコード化する装置。

41. もし、上記第1のコード化されたイメージから生成された上記デコードされたベクトルが、上記現在のイメージと比較して大きな平均2乗誤差を有し、上記前のコードブックの項目を更新する手段が動作可能であれば、上記更新する手段は上記現在のコードブックとして用いられる新しいコードブックを生成する手段を更に含む請求項40に記載の現在のイメージをコード化する装置。

42. もし、上記項目更新手段が少なくともN個のイメージを順番にコード化するように動作可能であれば、上記新しいコードブックを生成する手段が活性化される請求項41に記載の現在のイメージをコード化する装置。

43. 上記更新手段が、上記前のコードブックに対するベクトルを表すツリーのノードをたどって行き、上記コードブックの最良の項目を決定し、現在のコードブックで用いる手段を含む請求項35に記載の現在のイメージをコード化する装置。

44. 上記現在のイメージのベクトルと比較して、最悪の誤差を有する上記ツリー内の項目を除去する手段を更に含む請求項43に記載の現在のイメージをコード化する装置。

45. 上記ツリー内で最良の項目を決定する上記手段が、現在のイメージのベクトルと上記ツリーの間の平均2乗誤差を決定する手段を含む請求項43に記載の現在のイメージをコード化する装置。

46. 上記ツリー内に追加の項目を作成し、現在のイメージをより良く表す手段を更に含む請求項43に記載の現在のイメージをコード化する装置。

47. 追加の項目を作成する上記手段が上記ツリー内でノードを分割し上記現在のコードブックに対する2つの新しい項目を作成する手段を含む請求項46に記載の現在のイメージをコード化する装置。

48. 上記ツリー内でノードを分割する手段が、上記ノードがそのノードに関連する現在のイメージからのベクトルの閾値より大きい個体数を有するという判定によって動作状態となる請求項47に記載の現在のイメージをコード化する装置。

49. 上記ツリー内でノードを分割する手段が、上記ノードが閾値より大きいベクトル歪を有するという判定によって動作状態となる請求項47に記載の現在のイメージをコード化する装置。

50. 関連する現在のイメージからのベクトルを持たない項目を上記ツリーから除去する手段を更に有する請求項44に記載の現在のイメージをコード化する装置。

51. a. n 個のイメージから生成された N 個のイメージを N 個のコード化されたイメージとしてコード化する手段であって、 n は N と等しいか又は小さい上記コード化する手段と、

b. 上記 N 個のイメージのコード化されたイメージが上記 N 個のイメージのそれぞれの良好な近似であるかどうかに基づいて N を決定する手段と、
を備えた N 個のイメージをコード化する装置。

52. n が1である請求項51に記載の N 個のイメージをコード化する装置。

53. n が N と等しい請求項 51 に記載の N 個のイメージをコード化する装置。

54. 上記決定する手段が、上記コード化されたイメージのそれぞれから生成された各デコードされたイメージとそれに関連するコード化されないイメージの間の平均 2 乗誤差が閾値より大であるかどうか判定する手段を含む請求項 51 に記載の N 個のイメージをコード化する装置。

55. 上記閾値が、前のイメージと関連するコード化されないイメージの平均 2 乗誤差を含む請求項 54 に記載の N 個のイメージをコード化する装置。

56. 上記前のイメージが、前のコード化されないイメージのフィルタリングされた表現である請求項 55 に記載の N 個のイメージをコード化する装置。

57. 上記前のコード化されないイメージの上記フィルタリングされた表現がフィルタ手段によって生成され、上記フィルタ手段はレート制御メカニズムの制御下にある、上記レート制御メカニズムは適合的に上記フィルタ手段を調整し、所望のレートでコード化されたビット・ストリームを生成する請求項 56 に記載の N 個のイメージをコード化する装置。

58. 平均のブロック平均 2 乗誤差より大きい平均 2 乗誤差を有する N 個のコード化されたイメージに対する第 1 のブロック数が、誤差ブロックの閾値より大であるかどうか判定する手段と、更に上記判定する手段に応答して $N+1$ 番目のイメージに対して新しいコードブックを生成する手段を含む請求項 55 に記載の N 個のイメージをコード化する装置。

59. 上記コードブックを更新する手段であって、上記更新する手段は、上記 N 個のイメージの現在のイメージからブロックを計算し、上記コードブックの項目を更新して上記現在のコードブックを生成する手段を含む請求項 51 に記載の N 個のイメージをコード化する装置。

60. 上記現在のコードブックを用いて、上記現在のイメージのコード化されたイメージから生成された、デコードされたイメージがフレームの閾値より大きい平均 2 乗誤差を有するかどうか判定する手段を有する請求項 59 に記載の N 個のイメージをコード化する装置。

6 1. 上記判定する手段の活性化に応答して現在のイメージから置き換えコードブックを生成する手段を更に含む請求項6 0に記載のN個のイメージをコード化する装置。

6 2. a. 初期のイメージから生成されたベクトルを表すベクトルを有するコードブックを受け取るステップと、

b. 上記一連のイメージの次のイメージを検索し、上記コードブックを用いて上記次のイメージを次のコード化されたイメージとしてコード化するステップと、

c. 上記次のコード化されたイメージから生成された次のデコードされたイメージと上記次のイメージとの間のフレームの平均2乗誤差が上記初期のイメージから生成された初期のコード化されたイメージから生成されたデコードされた初期イメージに対する前のフレームの平均2乗誤差より大であるかを判定し、もし

そうなら、上記次のイメージから新しい現在のコードブック全体を生成し、上記新しい現在のコードブックを用いて上記次のイメージをコード化するステップと、

d. そうでない場合、上記次のイメージの関連するコード化されないブロックより大きな平均2乗誤差を有する上記次のデコードされたイメージのデコードされたブロックのピーク数がブロックの閾値を超えれば、上記次のイメージから新しい現在のコードブック全体を生成し、上記新しい現在のコードブックを用いて上記次のイメージをコード化するステップと、

e. そうでない場合、上記コードブック内で最悪の項目を決定し、更新されたコードブックの項目を生成し、上記次のイメージから新しい現在のコードブックを生成し、上記現在のコードブックを用いて上記次のイメージをコード化するステップと、

から成る一連のイメージをコード化する方法。

6 3. 上記コード化するステップが、上記次のイメージからのベクトルを上記コードブックの項目に関連づける請求項6 2に記載の一連のイメージをコード化する方法。

64. 上記コードブックが、上記次のイメージから生成されたコードブック・ベクトルを有するツリーを含む請求項63に記載の一連のイメージをコード化する方法。

65. 上記更新するステップが、上記次のイメージからのベクトルを持たない上記コードブックから、項目を除去するステップを含む請求項64に記載の一連のイメージをコード化する方法。

66. 上記更新するステップが、上記コードブック内で最も歪んだ項目を決定するステップを更に含み、上記最も歪んだ項目のそれぞれに対して：

- a. 上記歪んだ項目のそれぞれに対して子供の項目を作成し、
- b. 上記最も歪んだ項目のそれぞれに関連するベクトルを上記子供項目に関連づけるステップを含む請求項65に記載の一連のイメージをコード化する方法。

67. 上記コードブックから、上記最も歪んだ項目のそれぞれを除去するステップを更に含む請求項66に記載の一連のイメージをコード化する方法。

68. 変更された上記コードブックの項目をデコード手段に伝送するステップ

を更に含む請求項67に記載の一連のイメージをコード化する方法。

69. 上記項目が n 個の項目を有し、更新された上記現在のコードブックが N 個の項目を有し、 n は N より小さく、上記コードブックの n 個の項目は、コード化された次のイメージから生成されたデコードされた次のイメージと上記次のイメージの間の平均2乗誤差に従って導かれる請求項62に記載の一連のイメージをコード化する方法。

70. a. イメージから生成されたベクトルを表すベクトルを有するコードブックを受け取るステップと、

b. 上記一連のイメージの第1のイメージを検索し、更新されたコードブックを生成して現在のコードブックを生成し、上記現在のコードブックを用いて上記第1のイメージを第1のコード化されたイメージとしてコード化するステップと、

c. 上記第1のコード化されたイメージと上記第1のイメージの間の平均2乗誤差が、前のコード化されたイメージと関連するコード化されないイメージに対

する前の平均2乗誤差より大であるかを判定し、もしそうなら、現在のコードブック全体を生成し、上記現在のコードブックを用いて上記第1のイメージを第2のコード化されたイメージとしてコード化するステップと、
から成る一連のイメージをコード化する方法。

71. 上記コード化するステップは、上記第1のイメージからのベクトルを上記コードブックの項目に関連づける請求項70に記載の一連のイメージをコード化する方法。

72. 上記コードブックは、上記イメージから生成されたコードブック・ベクトルを有するツリーを含む請求項71に記載の一連のイメージをコード化する方法。

73. 上記更新するステップは、関連する上記第1のイメージからのベクトルを持たない上記コードブックの項目を除去するステップを含む請求項72に記載の一連のイメージをコード化する方法。

74. 上記更新するステップが、上記コードブック内で最も歪んだ項目を決定するステップを更に含み、上記最も歪んだ項目に対して：

a. 上記歪んだ項目のそれぞれに対して子供の項目を作成し、

b. 上記最も歪んだ項目のそれぞれに関連するベクトルを上記子供項目に関連づけるステップを含む請求項73に記載の一連のイメージをコード化する方法。

75. 上記コードブックから上記最も歪んだ項目のそれぞれを除去するステップを更に含む請求項74に記載の一連のイメージをコード化する方法。

76. 変更された上記コードブックの項目をデコード手段に伝送するステップを更に含む請求項75に記載の一連のイメージをコード化する方法。

77. 上記項目がn個の項目を有し、更新された上記現在のコードブックがN個の項目を有し、nはNより小さく、上記コードブックのn個の項目は、コード化されたイメージと上記イメージの間の平均2乗誤差に従って導かれる請求項70に記載の一連のイメージをコード化する方法。

【発明の詳細な説明】**改良されたベクトルの量子化****発明の背景****1. 発明の分野**

本発明はビデオの圧縮、圧縮解除に関連し、特にイメージの事前処理及びベクトルの量子化（VQ）を用いた改良されたビデオ圧縮／圧縮解除に関連する。

2. 関連技術の背景

マルチメディアや完全動画ビデオを必要とするその他のアプリケーションのような近代的アプリケーションは、ビデオ情報の記憶、伝送及び表示に費やされる処理帯域幅を低減するビデオ圧縮の標準の開発を必要とした。これは、高解像度のフル・イメージのビデオ情報を表す伝送、記憶のための大量のデータのためである。一般に図1 a、1 b及び1 cに示すような装置は、ベクトルの量子化技術に基づく入力イメージの圧縮、圧縮解除を行うために用いられる。例えば、図1 aに示すように、イメージ100は、冗長度を低減するため、あるいは入力イメージ100に含まれるデータ量を低減するため、入力イメージ又は一連のイメージに対する空間的又は時間的事前処理を適用するエンコーダ101に入力される。エンコーダ101は元のイメージ100よりも実質的に小さい圧縮されたイメージ102を生成する。ある種の従来技術システムでは、エンコーダ101は入力イメージ100の画素パターンをマッチングさせるために用いられるコードブック105を用い、画素パターンが圧縮されたイメージ102の中で別の画素パターンにマップされるようにする。このようにして、特定の色やグラフィックの情報を送らずに、イメージ内の各領域は、インデックスでコードブックのエレメントを参照することによってアドレスされる。ある種の従来技術のアプリケーションでは、圧縮イメージでの画質は損なわれるが、イメージ100から圧縮イメージ102へのイメージ・サイズの低減によって実質的節約が得られている。他の圧縮技術は「無損失（ロス・レス）」であり、追加の計算時間や大きなビット・ストリームというコストはかかるが、一般にデコードされたイメージに画質の低下はない。

逆に、圧縮解除されたイメージ132を生成するため、図1 bに示すように圧

縮されたイメージ102がデコーダ131に加えられる。再び、デコーダ131はコードブック105を用いて圧縮されたイメージ102に含まれるインデックスから、イメージ132に現れる画素パターンを決定する。デコーダ131はイメージをコード化するのに用いられた同じコードブック105を用いる必要がある。一般に従来技術のシステムではコードブックはコンピュータ・システムで表示するために圧縮又は圧縮解除されるイメージ又はイメージのセットに関連して固有である。

一般に105のようなコードブックはコードブック発生器152に加えられるイメージ又はイメージのトレーニング・セット151から生成される。コードブックは圧縮される1つ又は多くのイメージに対して特に生成され、そのコードブックは生成されたイメージをデコードするのに用いられる。コードブックは更に、将来コード化される一連のイメージの妥当な統計的表現である長いトレーニング・シーケンスに対して最適化することによって生成できる。このトレーニング・コードブックは大きな範囲のイメージ特性を表すと考えられる。このトレーニング・コードブックはしばしばエンコーダ及びデコーダで固定化されるが、コードブックの部分部分は順応して改良され得る。ある種の従来技術の体系では、コードブック発生器152及びエンコーダ101は一体となっている。コード化はコードブックの生成と同時に行われ、コードブックはトレーニング・イメージからでなくコード化されたイメージから導かれる。

図2はイメージのコード化及びデコードに対するベクトルとして知られる別々の領域にイメージ200がどのように区画化されるかを示す。1つの従来技術のアプローチでは、200のようなイメージは、「ベクトル」として知られる201及び202のような一連の2×2画素ブロックに分割される。201のような各ベクトルは4つの画素201a、201b、201c及び201dから成る。イメージがこのようなベクトルに分解されると、ビット・ストリーム内の各ベクトルを使って、(a)コードブックの生成を含むイメージのコード化を行い、(b)イメージのデコードを行う。イメージ200における201、202のような各ベクトルは、イメージ200を表すのに用いられる。イメージに含まれるベクトルの近似であるコードブックのエレメントを参照することによって1つのイ

メー

ジが表される。従って、201aから201dのような4つの別々の画素を用いてイメージを表す代わりに、ベクトル201に含まれる情報を近似するコードブック・インデックスを参照してイメージが表される。コードブック内の項目数によって、イメージ・ベクトルを参照するコードブック・インデックスを使用すると、実際の画素値201a-201dを用いてイメージを表すのではないため、ベクトルを表す記憶域を実質的に低減できる。

このような従来技術の装置は、図1aから1cを参照して論じたようにCODEC（コード化／デコード）として知られる装置に実施されており、これは対応するコードブックからの一連のイメージに対する圧縮されたビット・ストリームを生成し、コードブックを用いて後でイメージの圧縮解除を行う。例えば、このようなCODECは図3に装置300として示されている。CODEC300は2つの部分から成る。すなわちエンコーダ301とデコーダ351である。エンコーダ301は入力データ310としてビデオ、音声その他圧縮したいデータを受け取る。しかし、ビデオのコード化／デコードを論ずるこの出願の残余のために、同様の体系が他のタイプのデータにも適用できることを当業者は理解されたい。このような入力データはプリ・プロセッサ320に与えられ、コード化／デコードを簡単なタスクにするために特定のパラメータが調整されてデータを事前処理する。プリ・プロセッサ320は、イメージをある方法でコード化するためにベクトルの量子化を用いるベクトル量子化装置330に供給し、冗長度を等しく低減する。次にベクトル量子化装置330はパック／コード化処理340に出力し、更にビット・ストリームを圧縮する。レート制御メカニズム345は圧縮されたビット・ストリーム350のサイズに関する情報を受け取り、所望のデータ・レートを達成するため、様々なパラメータがプリ・プロセッサ320内で調整される。更に、プリ・プロセッサ320はコード化されたビット・ストリームをサンプリングし、画質のセッティングを調整する。

CODEC300は更に、コードブック再生器360を用いて圧縮されたビット・ストリーム350を受け取り、デコードするデコーダ351を含む。エンコ

ード内のデコーダは、イメージをデコードするためにパッキング340又はアンパッキング370の処理を行う必要はない。デコーダでは、コードブック再生器

360はアンパッキング処理370に供給され、完全なビット・ストリームに戻す。この処理の結果はポスト・フィルタ375に送られ、ディザリング380がイメージに対して行われ、最終的にイメージが表示される(390)。

従来技術の量子化処理の例は、次の文献に見られる: Gray, R. M. 「Vector Quantization」 (グレイ, R. M. による「ベクトルの量子化」 - IEEE ASSP Magazine 4-29 (1984年4月) (「グレイ」) 及び Nasrabadi, N. M. 「Image Coding Using Vector Quantization」 (ナスラバディ, N. M. による「ベクトル量子化を使ったイメージのコード化」) - A Review 「COMM-36 IEEE Transaction on Communication, 957-971 (1988年8月) (「ナスラバディ」)、このようなベクトルの量子化はツリー・サーチ (tree searched) のベクトル量子化装置の作成を含み、グレイの記事16-17頁及びナスラバディの記事75頁に記載されている。

コードブックの生成は反復的であり、計算機的に高価である。従って、フレーム毎にコードブックを必要とするいくつかの従来技術の方法では、コード化は低速となる。更に、トレーニング・シーケンスを用いる従来技術のシステムの欠点は画質であり、トレーニング・シーケンスのイメージと同様でない多くのシーケンスは受容できないであろう。全体的性能もまた気がかりである。いくつかの従来技術のテクニックは法外な処理を要し、リアル・タイムの圧縮を行うことができない上に許容可能な圧縮も達成されない。高速のデコード能力に対する需要は切迫しており、さもなくば、リアル・タイムの再生は不可能である。多くの従来技術システムは計算機的に高価なデコーダを有する。

発明の概要及び目的

本発明の1つの目的は、ベクトル量子化によってコードブックを効率的に生成し、イメージの空間的、時間的冗長性を低減する装置と方法及び圧縮システムの帯域幅を節約するためのイメージの関連処理を提供することである。

本発明の他の目的は、一般的従来技術のベクトル量子化技術に関連したエラー

を低減するため、イメージを効率的に区画化し、処理する手段を提供することである。

本発明の他の目的は、一般的従来技術のベクトル量子化技術に関連した計算を更に低減する手段を提供することである。

本発明の他の目的は、限定された帯域幅のチャネルでの円滑な再生に適合するため、圧縮されたシーケンスの結果的データ・レートを効率的及び効果的に制御する手段を提供することである。

本発明の他の目的は、圧縮されたデータのリアル・タイムのデコーディングを可能にする単純なデコード構造を提供することである。

本発明のこれらの目的は、ベクトルの量子化(VQ)の改良した方法と装置を提供し、データの圧縮のためのコードブックを構築することである。1実施例において、データはイメージ・データを含む。コードブック「ツリー」はN個の初期ノードを設定し、残りのコードブックをバイナリのコードブックとして作成することによって初期化される。子供のエン트리項目は最大歪、個体数等のような様々な属性の決定によって分割される。データから得られたベクトルは子供のノードに関連づけられ、次に代表的子供のエン트리項目が再計算される。この分割／再関連は反復して続けられ、前の子供と現在の子供の誤差の差が閾値より小さくなるまで行われる。この分割及び再関連は、ツリー内に最大数のターミナル・ノードが作成されるまで続けられ、合計誤差又は歪が閾値に達するまで又はその他の基準になるまで行われる。データは次にコードブックとコードブックを参照するインデックスから成る圧縮されたビット・ストリームとして伝送される。

図面の簡単な説明

本発明は例示によって示され、付随する数値に限定されるものではなく、類似の参照は類似の要素を示す。

図1 a - 1 c は、ビデオ・イメージを圧縮／圧縮解除するために用いられる従来技術のコード化／復号化装置を示す。

図2はイメージを2×2画素ブロックから成るベクトルに分割する従来技術の体系を示す。

図3は従来技術のCODEC（コード化／デコード）の機能ブロック図を示す。

図4は、無変化のブロックを識別するプリ・プロセッシング技術を示す。

図5a及び5bは、望ましい実施例で用いられるサブ・サンプリングの例を示す。

図6は望ましい実施例によって提供される改良したベクトルの量子化を用いて作成されるベクトル量子化ツリーを示す。

図7及び図8は、図6に示すツリーを作成するために用いられる改良したベクトルの量子化処理を示す。

図9a及び図9bは、「ゼロ」セルを除去し、残りのノードで反復することによって、ノードがベクトル・ツリー内でどのように更新されるかを示す。

図10は、望ましい実施例で用いられるビット・ストリームを示す。

図11-16は、図10を参照して論じられるビット・ストリームに含まれるデータの詳細図を示す。

詳細な説明

本発明はベクトル量子化の改良した方法に関する。以下の説明において、説明の目的で、本発明の完全な理解を与えるために、特定タイプのデータ、アプリケーション、データ構造、ポインタ、インデックス及びフォーマットが述べられている。しかし当業者には、本発明がこれらの詳細事項なしに実行できることは明白であろう。また、本発明を不必要にあいまいにしないため、既知の構造やデータはブロック図の形で示されている。

本発明の望ましい実施例は、図3に300として示される従来技術のCODECと類似の態様で構成される。これらはディスプレイ、プロセッサ及び様々な静的及び動的記憶装置を含む汎用のプログラムされたコンピュータ・システムで実施できる。これはまた、特別目的のアプリケーション用に設計された特別目的のビデオ用コード化又はデコード装置を含むことができる。もちろん、当業者には、望ましい実施例の方法と装置は、アプリケーションの要件に適合するように独立した論理装置、ファームウェア、アプリケーション固有の集積回路（ASIC

）又はプログラム論理アレイで実施できることが分かるであろう。

望ましい実施例は「C」プログラム言語のようなハイレベル・プログラム言語で実施され、汎用コンピュータ・システムで動作する。望ましい実施例を実施するために書かれたルーチンは実行可能なオブジェクト・コードにコンパイル及びアセンブルされ、ランタイムにシステムのプロセッサにロードされ、実行される。

本発明の議論は、ビデオ情報を参照して具体的に記載されているが、ここで論じた技術と装置は、ベクトル量子化を利用したオーディオの分野等他の分野にも同様の適応性があり、この出願のビデオ情報の議論が本発明を限定しているとみるべきでない。

プリ・プロセッシング

CODECから出力されるデータ・レートは、プリ・プロセッサ320を通してベクトル量子化処理に至る情報量を制御する。これは全体的及び局所的の2つのレベルで行われる。空間的解像度に対する全体的変化は、イメージの帯域幅を変える低域入力フィルタを入力イメージに対して適用することによって行われる。このフィルタの通過帯域幅は必要なデータ・レートにおける誤差で変わる。誤差が少なくなるにつれて、入力フィルタの帯域幅は増加し、CODECにより多くの情報が到着できるようにする。逆に所望のデータ・レートでの誤差が増加すると、入力フィルタの帯域幅が減少し、CODECに到達する情報を制限する。時間的解像度に対する全体的変化は、現在と前のフレームの間の差を判定することによって行われる。もし変化が閾値以下なら、現在のフレームはスキップされる。閾値はデータ・レート誤差から決定される。時間的帯域幅が減少する別の全体的メカニズムは2つのフレームの間の誤差の定義を拡張することによって、誤差計算の前にフレームの変形を可能にすることである。このような変形はパン（pan）やズームの保償を含むが、それらに限定されるものではない。

ベクトル量子化処理に到達することを許される情報量の局所的制御は、空間的サブ・サンプリングと時間的ブロック（又はより一般的には動きが保償されたブロックの局所的決定）を含む。望ましい実施例のシステムは、図3の330で示

す改良したベクトル量子化装置を実施し、コード化されるイメージのような非常に大きなベクトルのセットから代表的イメージ・ベクトルの小さなセット、コードブックという、を生成するのに非常に効率的である。このようなベクトル量子化装置によって生成されたコードブックからデコーダ 351 によって再構築されたイメージは、ある基準に関して元のものに近い。全体的圧縮／圧縮解除体系の性能は望ましい実施例において、ベクトル量子化装置の前にプリ・プロセッサ 320 によってビット・ストリームの内容を制御することにより更に向上する。このプリ・プロセッシングはベクトル量子化装置 330 に対して透明である。プリ・

プロセッサ 320 は、画質の損失を最小にしてイメージをコード化するのに用いられる情報量を実質的に低減する。望ましい実施例ではタグを用いて丁度その時に変わらないベクトルをコード化する代わりに明示する。ある閾値に従って変わらないため、「無変化」のブロックとして知られている。望ましい実施例では、ブロックは更に空間的サブ・サンプリングを用いて処理され、より良好な圧縮を達成する。更にプリ・プロセッサ 320 は、スピードを増大し又は画質を向上させるため、赤、緑、青 (RGB) 表すコード化から輝度とクロミナンス (YUV) を用いて表すコード化への変換等を行って、イメージ空間の特性を変えることができる。

無変化ブロック

望ましい実施例において、イメージ・ベクトルをコード化するか「無変化」ブロックのタグを送るかを決めるために一連の決定が行われる。「無変化」ブロックの場合、そのイメージ・ブロックに関してインデックスは送られる必要がないので、圧縮は殆ど常に向上する。コードブックを作成し、そのインデックスを見つけるイメージ・ベクトルが少ないので、コード化速度は向上する。前のフレームからのデコードされたブロックの上にスクリーン上で新しいブロックを配置しなくて良いため、デコード時間も向上する。従って、コードブックの要素を参照するインデックスを送る代わりに、プリ・プロセッサ 320 によって無変化タグが送られ、同一位置で、前のフレームのブロックからそのブロックが実質的に変わっていないことを示してベクトル量子化装置 330 を素通りする。これ

は図4を参照して示され、論じられる。処理400はステップ401で始まり、ステップ402でフレームNの次のブロックを取り出す。ステップ403でフレームNのこのイメージ・ブロックは次にプリ・プロセッサ320によって、デコードされたフレームN-1からの同一位置のイメージ・ブロックと比較される（デコードされたフレームN-1はエンコーダのビット・ストリームの出力から抽出されデコードされる。）。もしステップ404で検出されるように、2つのブロック間の誤差が閾値 μ より大きいと、ステップ406でそのブロックは変更されずにベクトル量子化装置330にコード化のために渡される。それ以外の場合、そのブロックはVQ330に対して「無変化」のタグが付けられ、ステップ

405に示すようにベクトル量子化は行われない。別の実施例では、無変化ブロックは前のフレームのブロックのうち、どれがサーチ領域内で良好な一致がとれたかを示す画素オフセットを持つことができることに留意されたい。

所望のデータ・レート及び画質が非常に高い場合、無変化のブロックとして μ をパスするイメージ・ブロックは、無変化ブロックとしてタグを付ける前に、より厳しいテストをされる。ブロックが無変化のブロック、ブロックの「エージ（age）」という、であるフレームの数は、チェックされ、最大許容エージを超えていないことが確かめられる。もし最大許容エージを超えていなければ、そのブロックは「無変化」ブロックとしてとどまる。もし最大許容エージを超えていれば、そのブロックと前のデコードされたフレームの同位置のブロックとの間の誤差がよりきつい閾値、例えば $\mu/2$ と比較される。これは無変化ブロックが所定の位置に長時間残っており、見る人から気付かれるのを防止するために行われる。ブロック・エージを用いる副作用は、多くのブロックがエージ化し、一緒に最大エージとなるときに起きる。これによって突然データ・レートが増大し、イメージの内容に無関係なデータ・レートの変動を起こすきっかけとなる。これを防止するため、望ましい実施例では各ブロックは様々な開始エージに初期化され周期的にリセットするようにしている。これはランダムに行うことができるが、もし連続したイメージの区画で行われた場合、エージ化はブロック・ヘッダでビット・ストリームをくずすことはあまりない。無変化ブロックのエージ化する主な

欠点は、データ・レートが高いことである。従って、所望のデータ・レートが非常に高い圧縮を要求せず、高い画質を要求するとき使用するのに適している。処理 400 は、ステップ 407 で決まるように、フレームが完全に処理されたときステップ 408 で終わる。ブロックに「無変化」としてタグを付ける決定は、一旦空間的サブ・サンプリングが行われると覆ることがある（例えばブロック・データが送られる）。もしデコーダに対して後続のブロックは「無変化」であることを知らせるためのブロック・ヘッダのオーバーヘッドによって、「無変化」のブロックを有する圧縮の正味の利益がなくなれば、「無変化」のブロックは先行するか後続するブロック・タイプに変更される。現在の実施例でこれが起こる例は、サブ・サンプルされたブロックの流れの途中で単一の 4×4 NC ($4 - 2 \times 2$ 無

変化) ブロックがあるときである。単一の 4×4 NC ブロックは先行する 1 つのヘッダと後続する 1 つのヘッダを必要とし、サブ・サンプルされたブロックの流れから分離し、ブロック・ヘッダ毎に 1 バイトとして 16 ビットを生ずる。もし単一の 4×4 NC ブロックがサブ・サンプルされたブロックに変えられたとすれば、単に 8 ビットのインデックス (256 のエントリ項目のコードブックに対して) を要し、伝送ビット数について云えば 4×4 NC ブロックにしておくよりも少ない犠牲で済む。処理 400 において無変化ブロック選択の決定に対して有用な様々な誤差と閾値の計算がある。望ましい実施例においてブロック比較に用いる誤差の基準は 2 乗誤差計算である。SNR (信号電力対ノイズ電力比) も別の実施例で用いることができ、高い輝度の領域に対する大きな誤差が許されるので有用である。これは人間の目が高輝度の領域における輝度の変化に鈍感であるという事実と関連する (ウェーバーの法則)。閾値 μ は望ましい実施例ではユーザの両質設定によって初期に決定されるが、レート制御要求と前の一連のフレームの平均 2 乗誤差 (frame-mse) に順応して初期値から変わり得る。望ましい実施例に用いられるアプローチは、無変化の閾値及び μ を次のように計算することである。

$$ncithreshfactor_n = ncithreshfactor_{(n-1)} + \beta * long_term_error_{(n-1)} (\beta=0.001)$$

$$\mu_n = ncithreshfactor_n * frame_mse_n$$

$$min\ mse\ bound < \mu < max\ mse\ bound$$

改善されたレート制御メカニズム 3 4 5 の議論で以下に詳しく論ぜられる long-term-error (長時間誤差) は、期間中の必要なデータ・レートを達成するベンチマークを提供する。もし long-term-error が、データ・レートは高すぎることを示すと、無変化のブロックはより頻繁にフラグが付けられる。逆に、もし long-term-error が、生成されたデータ・レートは所望値より低いということを示すと、無変化のブロックはそれほど頻繁にフラグを付けられない。瞬間的に反応する代わりに μ は β によってバッファされ、データ・レートを変える反応時間の時定数 (又は遅れ) を効果的に制御する。これは振動的データ・レートを防止し、完全にデータ・レートで駆動されるのではなく、多くのビットを生成する多くのバリエーション

ョンを有する複雑なイメージと、少ないビットを生成する少ないバリエーションを有するあまり複雑でないイメージに対する公差を許す。所与のシーケンスにおける達成可能な画質幅があるため、無変化の閾値 μ は frame-mse を考慮してシーケンスの直前にコード化された部分の画質を維持する。frame-mse はレート制御 3 4 5 によっても用いられ、これについてはレート制御の章で詳細に論ずる。

空間的サブ・サンプリング

望ましい実施例においてプリ・プロセッサ 3 2 0 によって行われる別の技術は空間的サブ・サンプリングである。サブ・サンプリングはベクトル量子化装置 3 3 0 によってコード化される情報量を低減するのに用いられる。これによっていくらかの空間的画質の犠牲で高速のコード化と高度の圧縮が得られる。主なチャレンジは高画質、高圧縮を維持することである。望ましい実施例で取り得る 2 つのアプローチがあり、それぞれ異なった利点がある。第 1 のアプローチでは、イメージは「スムーズ」と「ディテイル」の領域に分けられ、「スムーズ」のブロックはデータ・レートの要求に従ってサブ・サンプリングされる。例えば、「ス

ムーズ」領域は元のブロックと対応するサブ・サンプリング及びアップ・サンプリング (upsampling) されたブロックの間で平均2乗誤差を比較することによって決まる。これはサブ・サンプリングされた「スムーズ」領域は、通常最小の産物すなわち誤差を生成するので有利である。このアプローチに対する別の利点は、2つの別々のコードブックがサブ・サンプリングされた 2×2 C (「変化」) ブロックに関して生成され、各コードブックが数フレームに亘って共有されるときに発生する。「円滑性 (smoothness)」に完全に依存するサブ・サンプリングによって、2つのコードブックは「スムーズ」及び「ディテイル」の領域を多くのフレームに亘って表すことができる。これは「スムーズ」領域のイメージ・ベクトルは通常多くのフレーム間で類似しており、また同じことが「ディテイル」領域についても云えるためである。ゾーン (zone) が用いられる第2のアプローチでは、イメージ内のブロックの位置もサブ・サンプリングの決定に影響を与える。第2のアプローチの利点はイメージのどの領域がポスト・フィルタに行くのかをデコーダに対して効率的に通信する (ビットに関して) 能力を有することであり、サブ・サンプリングのブロックを奥めることによって、より効率的ブロック・ヘ

ッダのコード化の能力を有することである。サブ・サンプリングの処理は図5 aを参照して論ぜられる。サブ・サンプリングに関して、イメージは図5 aに示すように 4×4 のブロックに分割される。各 4×4 ブロックは、サブ・サンプリングとして選択されると510のような 2×2 ブロックに縮小される。望ましい実施例で行われるフィルタ・サブ・サンプリング動作は、4つの 4×4 画素ブロックのそれぞれの加重平均を用いて (例えば画素1-3、5-7、9-11及び17-23から成るブロック518) サブ・サンプリングされたブロック516 (ブロック518の場合画素1、2、5及び6のブロック) を表す。別の実施例では、図示のように単一の画素 (例えば1、3、9及び11) がサンプリングされ、より簡単なサブ・サンプリングの体系でサブ・サンプリングされたブロック510の代わりに用いられる。もしイメージ全体がこれらの技術の何れかを用いてサブ・サンプリングされると、改良されたベクトル量子化装置330に入るベク

トルの数は4のファクタで低減し、よって、最終ビット・ストリームのコードブック・インデックスの数もまた4のファクタで低減する。別の実施例では、サブ・サンプリングは水平方向にのみ又は垂直方向にのみ或いは、2以上のファクタで4×4画素以上のブロックを2×2画素ブロックにサンプリングすることによって各方向に行うこともできる。デコード中、改良されたデコーダ351は、インデックスに先行するヘッダにおいて、510のようなブロックの含まれるインデックスはサブ・サンプリングされたブロックを指示していることを検出し、520のような完全な4×4ブロックを再生するために各画素を水平、垂直の両方向に1つずつ模写する（例えば、4画素から成るブロック521を見ると、それぞれは単純サブ・サンプリングの場合における画素1と同じである）。ブロック521は4つの1の代わりに4つの γ で表され、 γ はブロック518の加重平均であることに留意されたい。別の実施例では、既存の画素の間の画素は良好な結果を得るため、相隣れる画素から補完することができる。しかしこれはデコーダの速度に有害な効果を与え得る。

「円滑性 (smoothness)」が判定される方法は、もし1つのブロックがサブ・サンプリングされる場合、どの程度2乗誤差が生ずるかに依存している。サブ・サンプリング動作は次の誤差計算に示されるようにフィルタリングも含む。2乗

誤差 ϵ は図5bに示す560（画素 $a_0 - a_3$ からなる）のような2×2ブロックのそれぞれとそれを囲む4×4ブロック555（画素 $a_0 - a_3$ 及び $b_0 - b_{11}$ からなる）の平均 γ との間で計算される。

$$\gamma = \frac{1}{16} \left(\sum_{i=0}^{11} b_i + \sum_{i=0}^3 a_i \right)$$

ブロック518から計算された γ は2×2ブロック521の画素1の値の代わりに用いられる。もし560のような2×2ブロックがサブ・サンプリングされると、それを囲む4×4 γ （ブロック555）の平均が4つの個々の画素値 $a_0 - a_3$ の代わりに送られる。平均 γ はブロックネス (blockiness) を低減するのに役立つ。従って図5を参照して示されるように、値 γ は、ブロック530の元の4つの画素値 $a_0 - a_3$ の代わりに送られる。次に2乗誤差 ϵ は加重係数 κ によつ

てスケーリングされ、人間のシステム輝度に対する感度に近似される（MSEの代わりにSNRをおおよその近似として用いることもできる）。従って、サブ・サンプリング誤差が同じであると仮定して高輝度の領域は容易にサブサンプリングすることができる。4つのスケーリングされた誤差は加算されて560のような各 2×2 ブロックに関連する誤差を生成する。

$$\epsilon = \sum_{i=0}^3 k[Y_i] * (a_i - \gamma)^2$$

Y_i : 画素 a_i の量子化された輝度値

サブ・サンプリングのための候補として 4×4 ブロック500を順位づけるため、 4×4 500の角に配置された4つの 2×2 ブロックからのサブ・サンプリング誤差 ϵ のそれぞれが加算される。レート制御が、所望のフレーム・サイズに合うように十分なブロックがサブ・サンプリングされたことを判定するまで、ブロックは最小の誤差ブロックから最大の誤差ブロックに向けてサブ・サンプリングのために選択される。別の実施例では、イメージの端がサブ・サンプリングされるのを防止するため、当業者には既知の端検出方法によってイメージの端が抽出される。サブ・サンプリングの決定をサブ・サンプリング誤差に依存するこ

とは、サブ・サンプリング及び端を越えるアップ・サンプリングが最大誤差を生ずる傾向があるため、多くの端を保護する傾向がある。しかし、端検出で見つかった端をはっきり保護することは、ある場合には有用である。

純粹に誤差に基づいたサブ・サンプリングは、多くの場合うまく動作するがサブ・サンプリングされたブロックが必ずしも互いに隣接して発生しないイメージがある。従って、サブ・サンプリングされないブロックの隣のサブ・サンプリングされたブロックの出現は、可視的に見ている人を悩ます閃光効果を起こし得る。いくつかのブロックはサブ・サンプリングされ、その他ブロックはサブ・サンプリングされないでブロックが動いているように見える。第2に、もしサブ・サンプリングされたブロックと標準的にコード化されたブロックが、空間的に混在すると、プリ・プロセッサ320によって識別されるブロック・ヘッダによってブロック・タイプの変化を表さねばならないので、かなりの帯域幅（ビットに

おける) が消費される (ブロック・ヘッダについては以下にビット・ストリームのシンタックスを参照して詳しく論ずる)。このようなイメージにおいては、別の実施例のコード化体系において、ゾーンを用いて誤差のみに基づくサブ・サンプリングする前述の2つの欠点を低減できる。イメージはプリ・プロセッサ320によって32の矩形ゾーン (水平8、垂直4) に分割され、それぞれは自らに関連した重みを有する。明らかに、ゾーンの数とそのサイズは全く多様である。

1実施例では、中心のゾーンをサブ・サンプリングされにくいように境界ゾーンのイメージの重み付けが行われる。このことは、カメラは主旨対象物の中心に向けられるため、見る人は端にあまり注意を払っていないと云うことを前提としている。別の実施例では速い動きを使って、サブ・サンプリングの産物を隠している。もし当業者に既知の運動予測アルゴリズムによって、動きが速くないと判定されると、動きの領域をサブ・サンプリングすることを難しくすることは有用である。このことは、見ている人は運動物体を追跡し、動きが速くなければサブ・サンプリングの産物に気付くということを前提としている。

望ましい実施例の第2のアプローチにおいて、ゾーンはゾーンの誤差、平均2乗誤差 ϵ に従って分類される。

$$\epsilon_j = \sum_{\text{zone } j} \epsilon$$

$$\hat{\epsilon}_j = \frac{\epsilon_j}{\# \text{ of subsampled pixels in zone } j}$$

各ゾーンはその位置に従って重み付けされ、ゾーン誤差ZEを生成する。

$$ZE_j = \hat{\epsilon}_j * \text{zone_weight}[j], \quad \text{zone } j$$

サブ・サンプリングのためにタグを付けられたブロックはゾーン誤差に関して最良ゾーンから最悪ゾーンの順にサブ・サンプリングされる。これはレート制御345によって要求されたサブ・サンプリングの数に達するまで行われる。改良されたデコーダ351は、入力ビット・ストリーム350から特定の基準で (画質設定等) どのゾーンがサブ・サンプリングされたかを判定でき、ブロックネス (blockiness) を柔らげるためにこれらのゾーンをポストフィルタ (処理375)

にかけるかどうか決める。サブ・サンプリングは帯状であるので、デコーダ351はイメージ全体をポストフィルタにかけないで、どこに努力を集中すればよいかを知っている。この情報をデコーダに送るのにオーバーヘッドは最小であり、32の矩形ゾーンの場合わずか32ビットである。ゾーン全体がサブ・サンプリングされるのを防止するため、edge-mseより小さい誤差を有するブロックだけがゾーン内でサブ・サンプリングされる。edge-mseの値はレート制御によって制御されるので、所望の圧縮フレーム・サイズが大きければ多くのブロックがサブ・サンプリングから保護される。

$$\text{edge-mse}_{n+1} = \text{edge-mse}_n + X * \text{long-term-error}$$

別の実施例ではedge-mseは重み付けることができ、当業者に既知の端検出方法で抽出されたイメージの端は、サブ・サンプリングから保護される。

方向性フィルタリング

空間的冗長度も、別の実施例における「方向性」フィルタリングを行うことによって端とディテール (detail) の最少の損傷で低減することができる。この処理は、画素を囲む領域に対して水平、垂直、上向き対角線、下向き対角線のフィ

ルタを実行し、最小誤差を生成するフィルタを選択する。もしフィルタの長さが3タップ (tap)、(フィルタ係数) であれば、図5aの画素6のフィルタをかけた値を計算することは、画素6に関するフィルタをかけた値を生成するため、画素5、6及び7に対して水平フィルタを適用し、画素2、6及び10に対して垂直フィルタを適用し、画素1、6及び11に対して下向き対角線フィルタを適用し、画素9、6及び3に対して上向き対角線フィルタを適用することを意味する。例えば、「水平フィルタ」を実行するために値は f_h であらわされ、 f_h は次のように計算される。

$$f_h = a_1 \cdot \text{画素5} + a_2 \cdot \text{画素6} + a_3 \cdot \text{画素7}$$

ここで、 a_1 、 a_2 及び a_3 は重み係数であり、それぞれ0.25、0.5及び0.25であり、3×3ブロックの中央画素に多くの重みが与えられ、結果の f_h は計算的に安価なシフト演算で計算される。これらのフィルタは3次元空間に適用でき、追加の次元は、別の実施例では、時間であることに留意されたい。

方向性フィルタの結果を比較することによって、イメージの端の方向付けも得られる。端の方向付けは、直交方向の対に関連する誤差の比を比較することによって抽出される。第1のステップは最小誤差min-directional-errorを生成した方向を選択することであり、この誤差を他の3方向のフィルタに関する誤差と比較することである。最小誤差のフィルタの方向に方向性の端があることを示す特徴は次の事項を含む。

- ・最小誤差を生成した最小誤差フィルタの方向と直交する方向
- ・最大誤差のフィルタは特にそれ自身と直交する方向と比較したとき、他の3方向よりも格別に大きな誤差を有する。フィルタをかけた領域が互いに他と非常に近い方向性誤差を有する場合、その領域は「無方向」である。「無方向」のブロックの領域は、再びその領域に対してフィルタをかけることができる。最小誤差のフィルタは、画素の周囲の特徴に従ってどの画素に対してもその特性を変えるので、非常に順応性がある。

YUV変換

望ましい実施例はまた、コードブックの生成とベクトル量子化装置330に関してベクトルの輝度とクロミナンス値(YUV)を用いて速度や画質を向上させ

る。YUV値は、ベクトルにおける画素の赤、緑、青(RGB)値からその再構築が計算的に安価な単純な変換を通して計算できる。例えば乗算の代わりにビット・シフトで実現できる次の変換である。

$$Y = \frac{R}{4} + \frac{B}{4} + \frac{G}{2}$$

$$U = \frac{R - Y}{2}$$

$$V = \frac{B - Y}{2}$$

ベクトル量子化装置330においてYUVを用いてコードブックの生成を行うことは、成分間のきつい動的範囲と相対的非相関性によってクラスタリングを向上

させる。従って画質の向上は顕著である。コード化速度が重要である状況では、クロミナンス (U, V) 値は2又は4でサブ・サンプリングされ、ベクトル量子化ステップ330において重み付け (例えばシフトで) られる。

望ましい実施例において、輝度及びクロミナンスは、入力イメージのベクトルのサブ・サンプリング又はフィルタリングのようなRGBの事前処理の後、プリ・プロセッサ320によってベクトル量子化装置330に送られる。別の実施例では、YUV変換は最初に行われ、サブ・サンプリングのような事前処理はYUV変換の後に行われる。如何なるレートにおいても、結果の事前処理データは改良VQ330にYUVのフォーマットで送られる。

改良されたベクトル量子化装置

ベクトル量子化 (VQ) はブロック又はデータのベクトルを表すのに効率的方法である。一連のデータ、画素、オーディオ・サンプル又はセンサー・データはしばしば各データを独立して取り扱うことによって量子化される。これをスカラ一量子化という。一方、VQはデータのブロック又はベクトルを量子化する。主なVQの問題は、データ・セットの許容し得る近似であり、コードブックと呼ばれる代表的ベクトルのセットを見つける必要があることである。許容性は通常元のデータ・セットと再生されたデータ・セットの間の平均2乗誤差を用いて測定

される。コードブック生成の一般的技術はLinde, Y, BuSo, A及びGray, Rによる「An Algorithm for Vector Quantizer Design (ベクトル量子化装置設計のアルゴリズム)」COM-28 IEEE Transactions on Communications (1980年1月) (LGBアルゴリズムとして知られている) に記載されている。コードブックを生成するためにLGBアルゴリズムを採用した技術は、コードブックの初期予測を生成するためイメージからの入力ベクトルをサンプリングすることから始まる。次に各入力ベクトルはコードブックの項目と比較され、最も近いコードブックの項目と関連させられる。コードブックの項目は各コードブックの項目と関連する平均ベクトルを計算し、現在の項目を平均ベクトルで置き換えることによって繰り返し更新される。次にコードブックが前回より大幅に向上したかどうかの判定が行われ、もし向上していなければ、入力ベクトルをコードブック

の項目と比較し、再関連させる等によって処理をくり返す。このコードブックの生成は大きなイメージ・シーケンスすなわちトレーニング・セットに対して行われるか、又はコードブックは各フレームに対して再生される。更に、この技術は効率向上のため、特定の従来技術のベクトル量子化システムに用いられるバイナリ・ツリーに適用できる。

改良ベクトル量子化装置330はツリー構造に編成されている。特定の従来技術の体系で用いられているバイナリ・ツリーではなく図6に示すようにツリーのルートでN個の子供ノード610が初期に生成される。これはいろいろなテクニックを用いて行うことができる。例えば、1実施例では、セグメンター (segmenter) を用いてイメージから代表的セントロイド (centroid) を抽出し、中心値を有するN個の初期ノードを生成する。別の実施例では、初期のセントロイドは、そのイメージ自身からN個のベクトルを抽出することによって、1つのイメージから決定される。従来技術のバイナリ・ツリーは単に2つの初期ノードの設定に依存している。バイナリ・ツリーには、2つの初期ノードにおける誤差がツリー内の残りのノードに伝搬するという欠点の悩みがある。望ましい実施例では、N個のノードが用いられ、値Nはイメージの特性によって変わる。この利点は、多くの初期ノードがルート・レベルでの間違っただビンニング (binning) のチャンスを低減するという事実に関連している。良好な画質と速い収束は、ツリー作成

においてN個の初期ノードを用いることから達成され、Nはイメージに適合し、通常2よりも大きい。

イメージに対して行われる改善されたベクトル量子化処理700は図6、7及び8を参照して示され、論ぜられる。N個の初期ノードの作成は図7のステップ702で行われる。ツリーの最上層610は、ステップ703で初期ノードの値をくり返し調整し、ベクトルをそれらに関連させることによりN個の初期ノードから改善される。この繰り返し処理は、繰り返しのノード・ビンニング／再計算処理を示す図8を参照して以下に記載されている。ステップ704で、最悪の歪を有するノードが決定され、その歪はノードのセントロイド値とその関連ベクト

ルの間の比較から計算される。望ましい実施例において、ノードに関連するベクトルとノードのセントロイド値の間の平均2乗誤差は歪の尺度として用いられる。どのノードが最も歪んでいるかの判定は、別の実施例で多くの尺度を用いて行われることに留意されたい。これらの尺度は、個体数、ノードに関する合計、ノードに関する平均歪あるいはノードに関するピーク歪が含まれる。どのレートにおいても、ステップ704で一度最大歪ノードが決定されると、ステップ705でこのノードは2つの子供ノードに分割される。もちろん、望ましい実施例では2つの子供ノードが記述され、用いられるが、別の実施例では2つ以上の子供ノードが作成されても良い。最良の代表ベクトルを得るため、ステップ706で子供ノードに対する繰り返し処理が行われる。この処理は、図8を参照してより詳しく説明されている。最も歪んだノードから作られた子供ノードに適用されたステップ703又はステップ706のような繰り返し処理は図8に示されている。この処理はステップ801で始まる。ステップ802で、親ノードと関連するベクトルのグループから図6の670に示す子供ノードに、代表的セントロイドを割り当てる。ルート・ノードの場合は、イメージの全ベクトルを用いて代表的セントロイドを作成する。次に各ベクトルは最も近いセントロイドを有するノードと関連づけられる（ビンニング）。次にステップ804で、各セントロイドと関連するベクトルとセントロイド自身との間の誤差が決定される。誤差計算は色々なテクニックを用いて行われるが、望ましい実施例においては平均2乗計算が用いられる。ステップ805で1度誤差計算が決まると、誤差の変化が特定の閾値以下

になったかどうか判定される。ステップ806で、ステップ803からのノードと関連するベクトルから新しいセントロイドが計算され、これはステップ803からの全てのノードに対して行われる。706に示す処理の最初の繰り返しでは、誤差の変化は非常に大きく、大きなプリセットの値から、計算された誤差値に進んでいく。しかしステップ803から806のループの次の繰り返しでは、誤差の変化は小さくなり、最終的に閾値より小さくなる。もし現在分割されているノードに関する誤差の合計が、ステップ805で決まるように、閾値より小さ

くなければ、ステップ806で新しいセントロイドが再計算され、処理703（706）が続いて行われ、ステップ803から806が必要に応じて再びくり返される。これは、ステップ805で検出されるように、誤差の変化が所定の閾値より小さくなるまで行われる。ステップ805で検出されるように、誤差の変化が閾値より小さくなると、処理703（706）はステップ807で終了し、図7の処理700に戻る。

この反復処理が図7のステップ707で完了すると、ツリー内で所望の数のターミナル・ノードが作られたかどうか判定される。ノードが分割される度に、2つ又はそれ以上の子供ノードがVQツリー600に生成される。従って望ましい実施例では、要求されるターミナル・ノードの合計数は、VQツリー600のノードが何回分割されるかを決定する。処理700は、必要なターミナル・ノードの数がツリー内に作られるまでステップ704から707を続ける。一度必要な数のターミナル・ノードが作られると処理700はステップ708で完了し、コードブックは出力ビット・ストリーム上に伝送され、図3に示すパック／コード化装置340に送られる。

望ましい実施例でノードに対して用いられるタイプ構成は次のような「C」プログラム言語で定義される。

```
typedef struct model {
    unsigned long *centroid:      // このノードに関するセントロイド
                                   // に対するポインタ
    unsigned long *vert-index-list: // このノードに関するベクトル・イン
                                   // デックスのリストに対するポイン
                                   // タ
    unsigned long *num-vect:      // このノードに関するベクトル数
    unsigned long distortion:     // このノードに関する合計歪
    unsigned long avg-dist:      // このノードに関する平均歪
    unsigned long peak-dist:     // このノードに関するピーク歪
    unsigned long percent-dist:  // このノードに関する歪のパーセン
```

ページ

```

unsigned long num-children :    // 子供の数
unsigned long ic-method :      // このノードの初期化方法
struct mode **children :       // このノードの子供ノードに関する構
                                造のリストに対するポインタ
struct mode *parent :          // このノードの親に対するポインタ
unsigned char terminal :        // これがターミナル・ノードかどうか
                                を示すフラグ
unsigned long *childrencptrs :  // 次に対するポインタのアレイに対す
                                るポインタ
                                // 子供のセントロイド（歪計算を単純
                                // 化し高速化するために用いられる）
}

```

600のようなツリーVQを構成するノードは、それぞれが上で定義したようなデータを有し、色々な歪尺度、ビンニングされたベクトル数、子供の数などのノードに関する特定の情報を維持する。この情報は上で論じたツリーの生成に役立つ。

600のようなVQツリーの生成に対する望ましい実施例のベクトル量子化処理700は多くの新しい技術を用いて実行される。

第1に、順応収束閾値（805で使用）は繰り返しの数を制御し、コードブック・ツリーを生成するのに用いられる。これは2つの方法のうちの1つで動作する。

1. もし完全なツリーが更新されることになれば、ゆるい収束基準が初期のN個のノードに適用される。完全なツリーは場面の変更が起こったか又はイメージが前のイメージから著しく変わった場合に更新される必要がある。

2. もし前のツリーからのルート・ノードが現在のツリー構築に用いられるとすれば、そのルート・ノードには繰り返しは行われない。ルート・ノードは類似のイメージ・シーケンスがコード化され、場面変化が検出されない場合再使用で

きる。従って610のようなN個の初期ノードは、前のフレームのVQから再使用できる。

第2に、望ましい実施例では、再生イメージの画質向上のため、修正距離尺度が用いられる。通常イメージ・ベクトルとコードブック項目の間の平均2乗誤差(mse)が用いられて所与のベクトルに対して一番近く一致するコードブック項目を決定する(例えば、図8のステップ803)。ツリー生成の初期の段階では、望ましい実施例はこの計算を修正して大きな誤差には2乗誤差より重い重み付けを行う。このようにして、大きな誤差には小さな誤差よりも多く重み付けがなされる。

第3に、複数の基準を使ってどのノードが分割されるべきかを決定する。採用される尺度には次のものが含まれるが、これらに限定されない。

1. 特定のノードに関する合計歪。
2. 特定のノードに関する平均歪。
3. 特定のノードに関する個体数。
4. 特定のノードに関する歪のパーセンテージ。
5. 特定のノードに関する最大歪。
6. 特定のノードに関する最大歪と最小歪の比。

ノードに関する合計歪は望ましい実施例で用いられる。しかし、別の実施例のツリー生成の最終段階では、もし個体数を尺度として用いれば良好な画質の結果が達成できる。もし平均2乗誤差が歪尺度として用いられれば、歪の合計は平均2乗誤差の合計である。他の歪尺度又はそれらの組み合わせの使用は、他の実施例で用いることができ、それぞれは、イメージの内容や所望の画質に応じて特定の利点を有する。

第4に、ノードを分割するのに複数の試行が行われる。たまたま特定のノードを分割する試行が失敗する。この場合、多くの他の初期条件が生成され分割がうまく行くように支援する。例えば、これが行われる1つの方法は、初期の分割に対してノイズを加えることである。平坦な、又は非常に滑らかに変化するカラー又は輝度の領域で特徴づけられる特定のイメージについては、ノードの分割は難し

い。少量のノイズが分割に先立ってイメージ・ベクトルに加えられる。ノイズは疑似ランダムであり、入力イメージ・データのゼロと2つの最下位ビットの間の範囲を有する。ノイズ生成の1つの態様は、疑似ランダム・ノイズ発生器を用いることである。この値がコード化される各ベクトルの各画素のRGB成分のそれぞれに加えられる。各画素のRGB成分のそれぞれに加えられたランダム・ノイズは分割を成功させるのに十分それらを区別させる。一般にどのノードを分割するか決定が行われたと仮定して、アルゴリズムは次のことを行う。

1. そのノードに関するベクトルのリストをサブ・サンプリングすることによって初期ノードのK個の候補を生成する。
2. これらの初期ノードを用いてベクトル・リストをクラスタリングする。
3. もしクラスタリングが失敗すれば（すなわち、全てのベクトルが1つのノードに集中する）、このノードをこの方法でクラスタリングするのに失敗したとして識別する。
4. このノードを分割する次の試行が行われるとき、ノード・セントロイドに対する異なった初期予測を用いる。この予測を生成するテクニックには次のものが含まれるが、これに限定されない。
 - a. 親ノードのセントロイドを乱れさす。
 - b. ノード・ベクトル・リストの中の最も歪んだベクトルを初期セントロイドとして捕らえる。
5. これらの初期ノードを用いてクラスタリングの試行が更に行われる。もし全ての方法がベクトル・リスト中で分割を生成するのに失敗すると、そのノードはターミナル・ノードとしてタグが付けられ、更に分割の試行が行われることはない。

第5に、コードブックの第1の層を複数フレームの間で再使用する。多くのイ

メージ・シーケンスにおいて、主なイメージの特徴は時間と共にゆっくり変わる（例えば、背景イメージは変わるかゆっくり動きがちである）。N個の初期ノードから成るコードブックのツリー610の最上層はこれらの特徴を捕らえる。計算速度と高画質に関する向上した性能は、1つのフレームから次のフレームに対

してツリーの最上層を再使用することによって得られる。この再使用はCODE Cで高いレベルからオーバーライドされる。例えば、場面変更の場合、これはCODE Cで検出されるのであるが、ルート・ノードが再使用されるよりもむしろ再生された方が高画質が達成できる。

第6に、コードブックの利用可能な項目を最高に使用するため、コード化の前にベクトルの平均値を除去するのが一般的である。これは良好な再生イメージの画質をもたらす一方、デコーダで追加の複雑さが発生する。望ましい実施例はデコーダの複雑さなしに平均剰余VQの多くの利点を与える技術を利用する。この技術は次のように動作する。平均値が大きなイメージ又は「ゾーン」に対して計算され、この平均値は大きなゾーンの全てのベクトルから減算される。残りのベクトルは通常のやり方でコード化される。デコーダでは大きなゾーンのそれぞれに対するコードブックは再構築される。これは、大きなゾーンの平均値を残りのコードブックに加えることによって行われる。この結果は、エンコーダでの大きなゾーンの数と同じ数のコードブックの生成となる。

可変サイズ、共用化、イメージに対する複数のコードブック

固定コードブックと適合性コードブックの組み合わせも別の実施例では可能であるが、望ましい実施例では各イメージはそのイメージの特性に適合したコードブックに関連しており、訓練された普遍的コードブックではない。別の実施例では、各イメージは正確に1個のコードブックまたはある固定サイズのコードブックを有するように限定される必要はない。別の実施例は可変サイズであり、複数のフレーム又は一連のフレームの間で共用でき、イメージをコード化する複数のコードブックを使用することを含む。これら全ての別の実施例において、利点は画質の損失を最小にして圧縮度を増大できることである。

可変サイズのコードブック

可変サイズのコードブックに関して、ツリー内のノードはある基準に達するまで分割され、これは指定した数のターミナル・ノードが存在する以前に起こる。1実施例ではコードブック・ベクトルの数は前のフレームから変化するブロックの数と共に増加する。言い換えれば、無変化ブロックの数が多ければ多いほどコ

ードブックは小さい。この実施例では、コードブックのサイズは明らかに絵のサイズに関係する。望ましい実施例で用いられる確固とした基準はフレームの平均2乗誤差（無変化ブロックは含まない）の維持に依存する。もし256個でなく128個の2×2コードブック・ベクトルが用いられれば、正味節約はそのフレームで768バイトである。この節約は、各2×2ブロックが輝度情報に関して画素毎に1バイトを含み、UとVのクロミナンス情報（YUV4:1:1の場合）に対しては2×2ブロック毎に1バイトであるので達成される。コードブック・ベクトルの数を256から128に減らすことによって $128 \cdot 6 = 768$ バイトの合計節約を生ずる。128のコードブック・ベクトルがMSEについて適切な画質を与えるイメージに関して、節約された768バイトはサブ・サンプリングされるブロックの数を低減するのに用いられ、従って見る人に対して画質を向上させる。

共用コードブック

望ましい実施例によって与えられる別の特徴は、共用コードブックの使用である。1つのコードブックを1つ又は多くのフレームに共用させることは、コードブックのオーバーヘッドを低減するために類似の内容を有するフレームを利用できる。共用コードブックの使用は、無変化のブロックを用いて効率的にコード化できない時間的相関関係を利用する。このような場合の例はパン（pan）されたシーケンスである。もし2つのフレームが1つの256エレメントのコードブックを共用するとすれば、節約は各フレームに128エレメントのコードブックを別々に使用させるのと等しい。しかし、もしフレームが完全に異なっていなければ画質は向上する。明らかに、別々に128エレメントのコードブックを用いる場合は、8ビットでなく7ビットのインデックスを用いることができるが、バイトの不整合によってビット・ストリームのパック／アンパックが扱いにくくなる。コードブックのオーバーヘッドの低減だけが共用コードブックの利点ではない。例えば、同一のコードブックを用いてイメージ間の時間的相関を増加すること

によって、時間的フリッカを低減できる。全く新しいコードブックがビット・スト

リームからアンパックされなくても良く、また各フレームに関してRGBに逆変換されなくても良いので、デコード速度の利益もある。

前のフレームから構築された共用コードブックがコード化しようとするフレームを良く表していることを確かめるため、共用コードブックは1つずつ新しいコードブックで置き換えることもできるし又は更新することもできる。第1に、フレームは共用コードブックを用いてコード化され、 frame_mse （元のフレームとコード化されたフレームの間の平均2乗誤差）が計算される。もし frame_mse が前のフレームからの frame_mse 又は前のフレームからの平均 frame_mse よりもあるパーセンテージで大きければ、共用コードブックは新しいコードブックで置き換えられる。 frame_mse がこのテストをパスしても、もしフレーム全体に関する平均MSEに対して、ある割合以上のMSEを有するブロックの数が、ある数以上（最悪のブロック）であれば、共用コードブックは置き換えることができる。この場合、エンコーダはコードブックに対する更新だけで最悪誤差のブロックを修復するのは難しいと仮定し、コードブック全体を再生する。別のやり方では、エンコーダは先ずコードブック更新リストの生成を選択し、最悪誤差のブロックがいくつあるかチェックし、もしある閾値以上の悪いブロックがあれば全く新しいコードブックを生成する。

ベクトル量子化の章で述べたように望ましい実施例は、共用コードブックの生成に用いたツリー構造を用いて共用コードブックを更新する。新しいフレームからの各イメージ・ベクトルはツリーのターミナル・ノードの1つと関連している（コードブックのベクトルと共に）。これは、ツリーのルートで始まり、どの子供が2乗誤差について近いか選択し、どの子供の子供が良く一致するかを選択する等によって達成される。イメージ・ベクトルは、このようにしてツリーのルート・ノードからターミナル・ノードへたどって行く。網羅的サーチを行えるけれども、網羅的サーチでなくツリー構造を用いてイメージ・ベクトルとコードブック・ベクトルとのマッチングをさせることによってコード化時間が向上する。更にツリー構造は、共用コードブックを更新するために、新しいノードを生成するのに役立つ。

コードブックの更新処理は数ステップを要する。まず、901（イメージ・ベクトルと関連しないコードブック・ベクトル）のようなゼロ・セルが見つけれられ、ツリー900から除去される。このブランチは図9aに示す。ゼロ・セルに関連するターミナル・ノード番号（コードブック・インデックス）が記録され、コードブックの更新は、ゼロ・セルであったコードブックの項目を置き換える。ツリー・ポインタは902が子供912及び913を指すように変えられる。これは図9aに変形されたツリーとして示されている。ツリーは次に、改良ベクトル量子化装置330に関して上で述べたように、ある基準で選択されたノード（図9b）（最悪の全体歪を有する n 個のノード）を分割する。これは図9bに920のツリーを930のツリーに変形して示されている。ゼロ・セルのために捨てられ（901）、又は分割によって親になったターミナル・ノードは、新しい更新されたベクトルで上書きされるようにタグが付けられる。最後に、ノード分割からの新しい子供は、上書きのためタグを付けたこれらのコードブックを上書きする。実際の上書きはデコーダで発生し、上書き情報はビット・ストリームを通して与えられる（以下を参照）。もしゼロ・セルがなければ、各ノード分割は2コードブック・ベクトル・スロットを要し、そのうちの1つは分割前のそのノードの親のものである。残りの子供は捨てたコードブック・ベクトルに対する単なる置き換えでなく追加のコードブック・ベクトルとして伝送される。

コードブックの共用によって、1つのフレーム又はフレームのセットから全体的に生成されたコードブックは、最大のコードブック・サイズ（例えば256）より小さいサイズ（例えば50%）にセットされ、追加のコードブック・ベクトルを共用コードブックを用いて追加できる。

別の分割及び置き換え方法は、前にターミナル・ノードであった親を置き換える必要がない。その代わり、2つの子供のうち1つが親と等しいと制約することによって、その親は置き換えられなくて良い。他の子供はゼロ・セルを置き換えるか又は追加のコードブック・ベクトルとして送られる。

複数コードブック

別の実施例では、別々のコードブックを各ブロック・タイプに対して生成することによって、又はそのイメージの異なった領域に対して別々のコードブックを

生成することによって、複数のコードブックを1つのイメージと関連づけることができる。前者は、圧縮の損失を最小（コードブックが共用の場合損失なし）にして画質を向上するのに効果的であり、後者は画質の損失を最小にして圧縮比を増大させるのに非常に効果的である。

別々のコードブックを用いてサブ・サンプリングされ及びサブ・サンプリングされていないイメージ・ベクトルをコード化することによって、従来技術のテクニックに比べていくつかの利点を得られる。独立したツリーが2つの異なったタイプのブロックの特性に対して特別に調整される。そのブロックはサブ・サンプリングされた領域には「スムーズ」となり、サブ・サンプリングされないブロックに対しては「ディテイル」となる。ブロック・タイプは空間的サブ・サンプリングの章で説明した誤差計算によって分けられる。「スムーズ」と「ディテイル」の領域の分離は、所望の圧縮がサブ・サンプリングを必要としないときに発生する。これは、「スムーズ」及び「ディテイル」のブロックが別々にコード化されているとき、別々のコードブックは非常によく動作するためである。各インデックスはブロック・タイプを通してコードブックと関連づけられており、コードブック・ベクトルの数はインデックス毎のビットを変えずに、又はVQのクラスタリング時間を増加せずに2倍にすることができることに留意されたい。これは、画質の顕著な向上をもたらす。更に、サブ・サンプリングされたブロックのコードブックと2×2Cブロックのコードブックは同じタイプの前のフレームのコードブックと共用できる。この場合、「スムーズ」領域と「ディテイル」領域を別々に維持し、数フレームに亘って各コードブックで一貫性があることが重要である。スムーズ及びディテイル領域への分離は、イメージのカテゴリに対して別々のツリーを定義するという一般的アイデアの特別の場合であることに留意されたい。カテゴリは、類似属性を有するイメージ内の領域を識別する分類子によって決められる。上で述べた簡単な場合には、2つのカテゴリ、スムーズ及びディテイルが用いられている。その他のカテゴリとしては端領域、テクスチャ及び平均値や偏差値など類似の統計を有する領域がある。

簡単に述べたように、複数のツリーがイメージ内の異なった領域と関連づけられる。これはコード化時間の短縮や圧縮比の増大に効果的である。例えば、粗い

グリッド（等しいサイズの8個の矩形）は8個の16エレメント・ツリーでコード化される。最悪誤差の矩形領域は、次に再び分割され、各矩形領域の各半分が16エレメント・ツリーを用いる。これは16個の矩形、従って合計256のコードブック・ベクトルになるまで続けられる。各インデックスは8ビットでなく4ビットを用いてコード化され、追加の2:1圧縮を与える。もしイメージが16個の固定した初期領域に分割され、それ以上領域の分割がなければ、コード化計算時間は大幅に短縮される。このテクニックは特に低画質、高圧縮、高速コード化モードに適応する。イメージの小片に対して、多くの小さなコードブックを用いることとイメージ全体に対して1つの256項目のコードブックを用いることとの間の妥協は、画質がそれ以上悪くならない画質を維持し、もっと多くの圧縮を得るのに効果的である。このような妥協では、非常に均一でわずかなコードブック・ベクトルしか必要としないイメージの部分にのみ小さなコードブックが用いられ、正規の256項目のコードブックは残りのイメージに用いられる。もし小さなコードブックに関連するイメージの部分が矩形であると考えられれば、小さなコードブックにスイッチするとき、デコーダに知らせるのに殆どオーバーヘッドを要せず、小さなインデックス（16項目のコードブックには4ビット又は64項目のコードブックには6ビット）となる。各コードブックに関連する領域が矩形であると考えられなければ、類似の画素を1つの領域にグループ化する当業者に知られた区画化テクニックで画質を向上できる。

レート制御

レート制御345は圧縮された素材が限定された帯域幅チャネルでデコードされるとき、改良ビデオ圧縮システムの重要なエレメントである。同期体系又はネットワーク又は電話線においてNフレーム/秒を維持するため、デコーダ351は1/N秒間に限定された帯域幅チャネルから1フレームのデータを読み、情報をデコードし、イメージをスクリーン上に表示しなければならない。レート制御345は最大フレーム・サイズを、ある数（アプリケーションによって異なる）以下に維持し、限定された帯域幅チャネルからデータを読むのに要する時間を短縮しようとする。これは2つのステップで行われる。すなわち、（1）データ・レートの観点から望ましいデータ・レートは何かを決定する。（2

) 画

質の要件 (ユーザによって定義されるか又はその他の方法で定義される) と共にこの望ましいフレーム・サイズを用いてコード化処理のパラメータを制御する。

レート制御体系は、望ましいフレーム・サイズが何であるかを過去のパフォーマンスと望ましいデータ・レートに基づいて決定する。target-frame-length (目標フレーム長) は次のように計算される。

$$\text{target_frame_length} = \frac{\text{desired data rate}}{\text{desired frame rate}}$$

現在のフレーム N に対する desired-frame-length (望ましいフレーム長) は、目標フレーム長から誤差項 frame-error を引いたものに等しく、あるフレーム数、例えばビデオ・データの 1 秒分の、平均である。

$$\text{desired-frame-length} = \text{target-frame-length} + \text{frame-error}$$

許容されるオーバーシュート又はアンダーシュートである frame-error は IIR (無限インパルス応答) フィルタとして帰納的に平均されることに留意されたい。これはまた、FIR (有限インパルス応答) フィルタとして別の実施例で実施できる。 α の値は、どれ位速く現在のフレーム誤差 (target-frame-length - avg-frame-length) が長期フレーム誤差に応答させるかに影響する。現在の誤差は target-frame-length といくつかのフレーム (例えば 1 秒分) の平均フレーム長 (avg-frame-length) として定義される。このレート制御体系は、望ましいデータ・レートを超えない過去 1 秒の平均データ・レートを維持する。フレーム・サイズのはらつきはフレーム単位で起こるが、これらのばらつきは平均効果で低下する。これらの関係は次のように決まる。

$$\begin{aligned} (\text{frame_error})_n &= (1-\alpha)(\text{frame_error})_{n-1} \\ &\quad + \alpha(\text{target_frame_length} - (\text{avg_frame_length})_n) \end{aligned}$$

$$(\text{avg_frame_length})_n = \sum_{i=n-k}^n a_i * \text{frame_size}_i$$

$$\text{where } \sum_{i=n-k}^n a_i = 1$$

desired.frame.lengthがNに対して決まった後、それはコード化パラメータ

(nctreshfactor及びedge-mse) に影響を与えるために用いられ、時間的フィルタリング及び空間的サブ・サンプリングが用いられる実施例において、そのパラメータはどれ位時間的処理と空間的サブ・サンプリングを適用するかを制御する。これらのコード化パラメータはユーザによって決められる空間的、時間的画質の好みによってセットされるが、それらは、システムがどの程度データ・レートの要求を維持しているかに従って画質設定に関する変動が許される。短時間にこれらのパラメータを大きく変動させることを許すよりも、それらは次のように計算される長期誤差を追跡する。

$$\begin{aligned} (\text{long-term-error})_n &= (1-\beta) (\text{long-term-error})_{n-1} + \\ &\beta (\text{target-frame-length}) - (\text{avg-frame-length})_n \end{aligned}$$

従って、long-term-error (長期誤差) に対する計算とフレーム誤差に対する計算の差異は α と β の差である。効果的であると決められた値は、 $\alpha = 0.20$ で $\beta = 0.02$ で、これらは望ましい実施例で用いられる。当業者は他の α 、 β の加重値が使えることが分かるであろう。

もしlong-term-errorが空間的サブ・サンプリング及び無変化ブロックに対するコード化パラメータ値を制御するのに用いられなければ、望ましいフレーム長が、どの程度データ・レートが維持されているかを追跡するのに用いられる。この場合無変化及びサブ・サンプリングの閾値はユーザの画質設定によってのみ決まることを前提としている。しかし、これはサブ・サンプリングと無変化のブロックがフレーム・サイズをdesired.frame.size (望ましいフレーム長) に短縮することを保証するものではない。この場合long-term-errorの値は、サブ・サンプリング及び無変化ブロックのパラメータ (nctreshfactor及びedge-mse) を変えることによって画質を低下させるのに用いられ、従ってデータ・レートを下げる。

コードブック・インデックスの伝送

イメージが改良された処理330によって、ベクトル量子化を通してコードブックに対するインデックスと関連づけられた後、ビット・ストリームは従来技術よ

り効果的にパックされ、将来の変化と両立する柔軟性を可能にし、余分なコード化オーバーヘッドを生成することなしに、イメージをデコードするのに必要な情報を通信する。インデックスは、それぞれコードブックに対するインデックス

又はコードブックのベース・インデックスからのオフセットとして伝送される。前者の場合、256項目のコードブックのどのベクトルが最も良くマッチするかを示すのに、イメージ・ベクトル当たり8ビットを必要とする。後者の場合、インデックス間の差は一般に256よりも大幅に小さいので、インデックス間に多くの相関があればより少ないビットでよい。イメージのある部分は互いに大きく離れたインデックスを有し、他の部分は強く関連するインデックスを有するので、通常2つの組み合わせが必要である。

図10を参照して示すように、ビット・ストリーム・シンタックスはシーケンス・ヘッダ1001) チャンク・ヘッダ1011、フレーム・ヘッダ1021及びコードブック・ヘッダ1012、1014を含む。コードブック・インデックスがこれらに続く。コードブック・インデックスは続くインデックスがどのブロック・タイプを参照するかを示すブロック・タイプ・ヘッダによって表される。2×2変化(2×2C)、2×2無変化(2×2NC)、4×4無変化(4×4NC)、4×4変化(4×4C)、サブ・サンプリングされた(4×4SS)、混合ブロックの異なった組み合わせ及び行画素ブロックは有用なブロック・タイプの例である。デコーダ351は、各イメージ・ブロックに対してどのコードブック・ヘッダを用いるか及びアップ・サンプリングを行うかどうかを知って、イメージを再構築することができる。ビット・ストリーム・シンタックスについて以下に論ずる。

シーケンス・ヘッダ1001はシーケンス全体に対する情報を伝達する。これらにはフレームの合計数、シーケンスがコードされたコーダのバージョン及びイメージ・サイズが含まれる。シーケンスは例えば動画全体を含む。単一のシーケンス・ヘッダ1001は一連のイメージに先行し、シーケンスについての情報を指定する。シーケンス・ヘッダ1001は殆どどんな長さでも良く、その長さを1つのフィールドに持つ。シーケンス・ヘッダに現在定義されているいくつかの

フィールドを図11に示す。シーケンス・ヘッダ1001はシーケンス・ヘッダID1101を含み、それによってデコーダがシーケンス・ヘッダであることを識別することができる。これはユーザに対してランダム・アクセスの再生を可能にするアプリケーションには有用である。更にシーケンス・ヘッダ1001は、

シーケンス・ヘッダ1001の長さを指定する長さフィールド1102を含む。次のフィールドはシーケンス内のフレーム数を指定するフレーム数フィールド1103である。これは整数値で符号なしの長語として格納され、シーケンスの長さが232個のフレームまで可能にしている。シーケンス・ヘッダの次のフィールド1104は、現在予備として確保されており、次の2つのフィールド、1105及び1106はシーケンス内のイメージの幅と高さを指定する。シーケンス・ヘッダ1001の最後のフィールドはバージョン・フィールドで、使用されるエンコード／デコード装置の現在のバージョンを指定する整数フィールドである。これは特定の特性を持っていたり、持っていなかったりする新しいシーケンスと古いシーケンスを区別するためのものである。これはシーケンスとエンコード／デコード体系の上位及び下位の互換性を可能にする。シーケンス・ヘッダはまた、イメージのシーケンスを指定するASCII又は文字列を含む（図示せず）。

図10に戻って、チャンク・ヘッダ1011は、共用コードブックが用いられているかどうかの、フレームの次のチャンク（chunk）についての情報を伝達するチャンク・タイプを持つ。チャンク・ヘッダはフレームのチャンクに対していくつのコードブックが用いられているかを指定することができる。チャンク・ヘッダ1011はフレームの「チャンク」に先行する。望ましい実施例において、チャンクは1つ又は多くのフレームで場面変化検出アルゴリズムのような装置によって他の「チャンク」から区別できるものである。別の実施例ではフレームのグループは、レート制御メカニズムのような別の技術を用いて関連づけられる。

2つのコードブック・ヘッダが図10のシーケンス例1000に示されており、これによって1フレーム当たり2つのコードブックの使用が可能となる。2つのコードブックを使用する例は、固定コードブック（フレームの「チャンク」に

対して静的)と順応性コードブック(フレーム毎に変わる)の使用である。コードブックのタイプとサイズは図13aに示すようにコードブック・ヘッダ1012と1014に含まれる。図10の1012又は1014のような各コードブック・ヘッダは、コードブック・タイプ・フィールド1301を有し、コードブック・タイプ、例えば固定か順応か、を指定する。コードブック・タイプはYUV(サブ・サンプリングされたUV又はサブ・サンプリングされないUV)、RG

B及びYUV更新コードブックを含む。その他のタイプは、本発明の精神と範囲の中で考慮されている。コードブックの「更新」については、コードブックに対する更新はコードブック・ヘッダに続いて送られる。コードブックのサイズはフィールド1302にバイト数で指定され、デコーダは、いつ次のフィールドが始まるかを検出できる。もしコードブック・タイプが「更新」コードブック(すなわち共用コードブック)なら、図13bに示す情報1013(又は1015)がコードブック・ヘッダ1012(又は1014)に続いて期待できる。この更新コードブックは、更新が必要なコードブック項目を識別するビットマップ1370を有する。このフィールドの後に、更新される各ベクトルに対するベクトル更新1371-1373が続く。このようにコードブック全体が再生成されるのではなく、選択された部分だけが更新され、更にデータ・レートの低減が行われる。もしサブ・サンプリングされたU及びVと共にYUVが用いられると、更新ベクトル1371-1373のそれぞれは6バイトから成り、4バイトはブロック内の各画素の輝度のためであり、1バイトずつがそれぞれUとVのためである。コードブックの更新について図9a及び9bを参照して説明した。

更にコードブックのオーバーヘッドを低減するために、1013及び1015のようなコードブックはYUV(輝度とクロミナンス)フォーマットに変形され、UとVは水平及び垂直方向(YUV4:1:1)に、ファクタ2でサブ・サンプリングされる。従って、コードブックはサブ・サンプリングされた情報を伝送することによりサイズが更に小さくなり、コードブックのサイズを2のファクタで低減する。

図12を参照して示すように、フレーム・ヘッダ1021はイメージ・サイズ

を幅フィールド1201と高さフィールド1202とに有し、フレーム・サイズの変動をいつでも可能にする。フレーム・ヘッダ1021はフレーム・タイプ・フィールド1203を有し、そのビット・パターンは、スキップ・フレームに対するヌル (null) フレームであるか、全体がサブ・サンプリングされたフレームであるか、キー・フレームであるか、又はフレームが他のフレームとコードブックを共用しているかを示す。他のタイプのフレームは、本発明の精神の中で考慮されている。サブ・サンプリングされたゾーン・フィールド1204は32ビット

トのビット・マップ・パターンであり、どのゾーンが（もしあれば）サブ・サンプリングされたかを示し、望ましい実施例では最大 2^{32} ゾーンまで可能にしている。

図14の部分1022に示すブロック・ヘッダは、デコーダ351に対してどのタイプのブロックがインデックスのセットに関連しているか及びいくつかのインデックスがそのセットにあるかを知らせる。これは図14を参照して示される。ヘッダ1401の最初の3ビットは、続くインデックスのセットが 2×2 Cブロック（変化ブロック）であるか、 4×4 NCブロック（無変化ブロック）であるか、 4×4 SSブロック（サブ・サンプリングされたブロック）であるか、混合ブロックであるか又は行画素値であるかを示す。もし最初の3ビットが、ブロック・タイプは混合でないことを指定すると、ヘッダ1401の最後の5ビットはいくつかのインデックス1402がブロック・ヘッダ1401に続くかを示す整数である。これは「ランレングス (runlength)」ブロック・ヘッダと呼ばれる。ブロック・ヘッダは 2×2 Cと 2×2 NCブロックの混合のような混合ブロックを指定する。この場合、長さのために確保されたヘッダの5ビットは、 2×2 Cと 2×2 NCブロックの混合のうち、いくつかの 4×4 がコード化されたかを指定する。別のやり方では、5ビットのうち1ビットは、もっと多くの混合の可能性を許すために用いられる。ビット・マップが続き、一番近いバイトにパッドされる。 2×2 C- 2×2 NC混合の例において、ビット・マップは「1」でブロック・タイプが 2×2 Cであることを、「0」でブロック・タイプが 2×2 NCで

あることを示す。ブロックは 4×4 の単位でも混合できる。もしビット・マップ・ヘッダがビット数をランレングス・ヘッダよりも少なくすれば、計算が簡単になる。「10010110101」のような一連の交互のブロック・タイプは、ビット・マップ・ブロック・ヘッダでうまくコード化され、1つのヘッダ・タイプが長く続くようなもの（例えば111111111000000000）はランレングス・ヘッダ・タイプでうまくコード化される。ブロックをより効率的にコード化するブロック・ヘッダが選択される。ビット・マップ・ヘッダは、頻繁に発生する短く続くブロックの効率的コーディングを可能にする。

「変化」ブロックの流れの中間において、「無変化」ブロックとタグを付けら

れたブロックの前後におかれるブロック・タイプ・ヘッダ1401の2バイトのオーバーヘッドのために、もし1行に少なくとも4個の 2×2 無変化ブロックがあれば、望ましい実施例のランレングス・ブロック・ヘッダは、ヘッダを有するインデックスの構造を乱すだけである。1410のようなヘッダを有するビット・ストリームにおいて区別するために、望ましい実施例のランレングス・ヘッダは4個の 2×2 NC（無変化）ブロックと一緒に発生し、1つの 4×4 NC（無変化）を作ることを要する。同じ位置の、前のフレームのブロックが代わりに用いられるので、次に続くN個のブロックが 4×4 NC（無変化）タイプであることを示す1410のようなブロック・ヘッダは、インデックスについてバイトを浪費する必要はない。デコーダ351は新しいイメージに対して、いくつかのブロックをスキップするかを知る必要があるだけである。実際の画素値又は特異な 2×2 ブロックが用いられるので、1402のような 2×2 Cブロックのインデックスは、4個のセットで発生する必要がない。もしある実施例で、実際の画素値又は特異な 2×2 C及び 2×2 NCブロックがサポートされていなければ、 2×2 Cブロックが4個のセットで発生することを仮定することは、1401のような 2×2 Cブロック・ヘッダと関連するブロック数を増加し、従ってブロック・ヘッダによる効果的オーバーヘッドを減少させる。例えば、もし 2×2 Cブロックがサポートされなければ、1つのブロックは8個の 2×2 C（変化）ブロックを識別し、それを8グループの $4 - 2 \times 2$ Cブロックの意味として解釈する。（2

—2×2Cブロックが2セットの4—2×2Cブロックとして解釈される図15、16の例を参照)。

更に、2×2Cブロックを参照する図14のインデックス1402は、4×4SSブロックを参照するインデックス1421と同じコードブックからである必要はない。このビット・ストリームの柔軟性は、256以上のコードブックを有し、バイト整合されないインデックス・サイズ(512のコードブック・ベクトルに対する扱いにくい9ビットのような)にジャンプすることなしに、非常に少ない圧縮の低下で高い画質のサポートを可能にする。

インデックス・パッキング

もしイメージ・ブロックがコードブック内で近接しており、類似のRGBカラー空間にあれば、ビット・ストリームに単にリストするのでなく、インデックスをコード化するとき、ベース・アドレスを用いるのが有利である。コードブック・ベクトルは「最悪」誤差のノードを分割して生成されるので、類似のイメージ・ベクトルがコードブック内に近く集まる傾向がある。似たイメージ・ブロックがイメージ空間で発生する傾向があるので(すなわちブロック間で空間的相関がある)、互いに近いインデックス値と一緒に発生する傾向がある。コードブック・インデックスの割り当てもまた、空間でのインデックスの差が最小となる方法で行うことができる。どのようにこれを用いてロスなしにビット数を減らすかの例が図15及び16に示され、説明されている。このパッキング処理は、図3のエンコーダ301の340で行われ、アンパックはデコーダ351の処理370で行われる。

図15で、もしコードブックが256の項目を持っていれば、ビット・ストリーム1500のコードブック・インデックスは、それぞれ8ビットを必要とする。言い換えれば、各インデックスはコードブックの元素に対して完全な参照を有する。上で論じたように、空間的相関のため、これらのインデックス値はベース・アドレスからのオフセットを用いることによって、もっと低減できる。これは図16に示されている。図16において、コードブック・インデックスのそれぞれは、もしインデックスが伝送されたベース・アドレスから、-8から+

7であるオフセットとして表されれば、わずか4ビットを要する。これはビット・ストリーム1600の1601に示される。ベース・アドレス1601は開始点として用いられ、1604のような現在のブロックのオフセット値は、現在ブロック1603に先行するインデックスにおける変化を参照することができる。ベース・アドレス・ヘッダ1601はベース・アドレスを定義して送られることを要し、差のコードが用いられる。大きく、可変であるコードブック・インデックス（コードブックの一端から他端）のセットを有する領域は、図15に示す完全なインデックスの伝送を用いて効率的にコード化され、ブロック・レベルで類似の領域は、図16に示す1600のようなビット・ストリームを用いてより効率的にコード化される。図16に示すように、ベース・アドレスからのオフセットを用いることは、元のインデックス値がベース・アドレスにオフセット値を加え

ることによって計算できるので、図15に示すテクニックと同様無損失である。

ビデオ・データを圧縮、圧縮解除する発明について記載した。この明細書において、本発明は図1から図16の特定の実施例を参照して説明された。しかし、添付請求の範囲に述べたように、本発明の広い精神と範囲から離れることなく様々な修正や変更がなされ得ることは明らかであろう。従って、明細書及び図面は、例示と考えるべきであり、限定と考えるべきでない。

【図1 a】

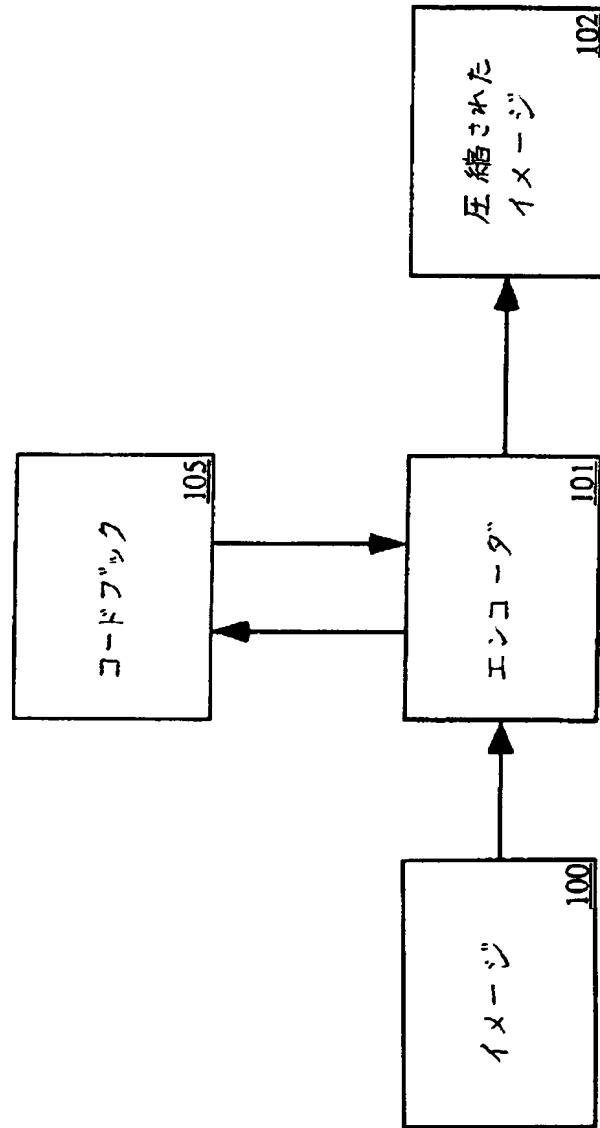


FIG. 1a (従来技術)

【図1b】

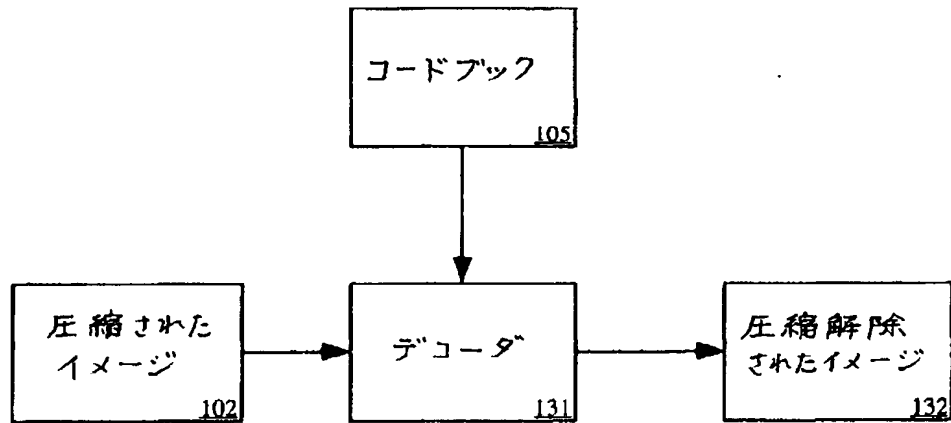


FIG. 1b (従来技術)

【図1c】

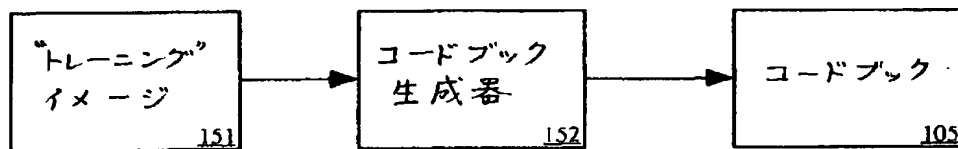


FIG. 1c (従来技術)

【図 2】

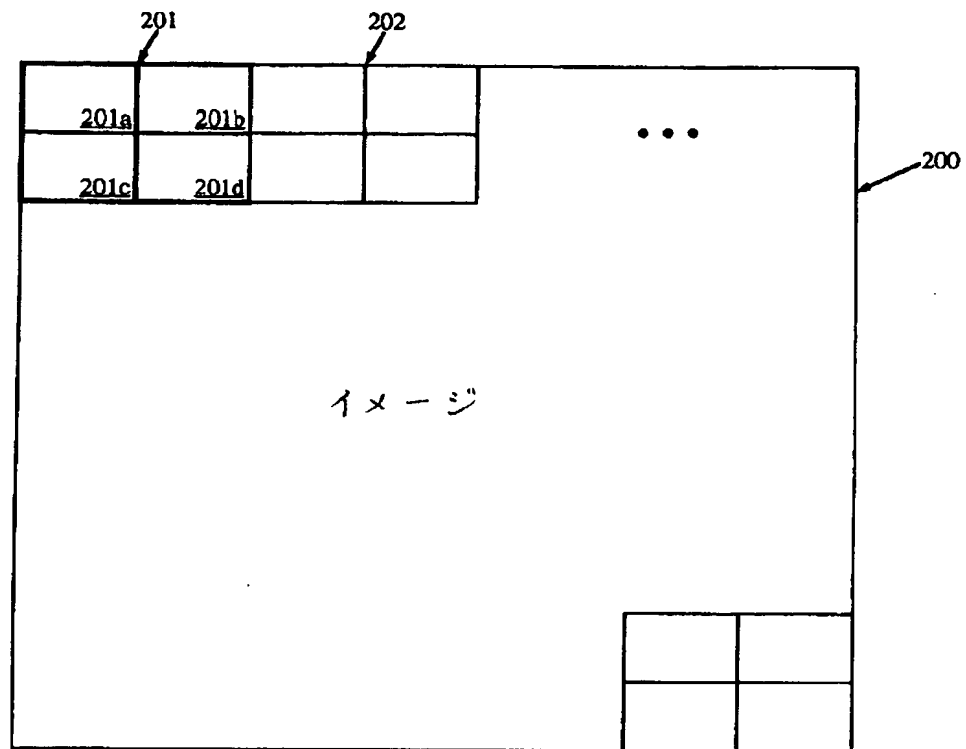


FIG. 2 (従来技術)

【図3】

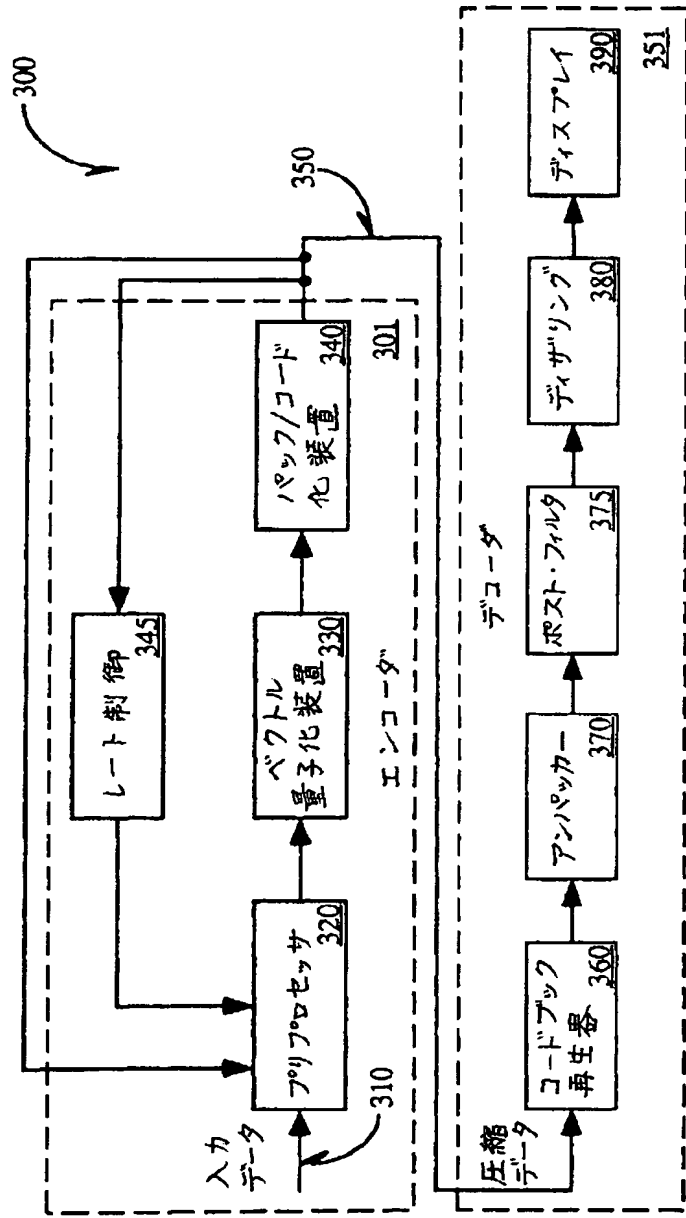


FIG. 3 (従来技術)

【図4】

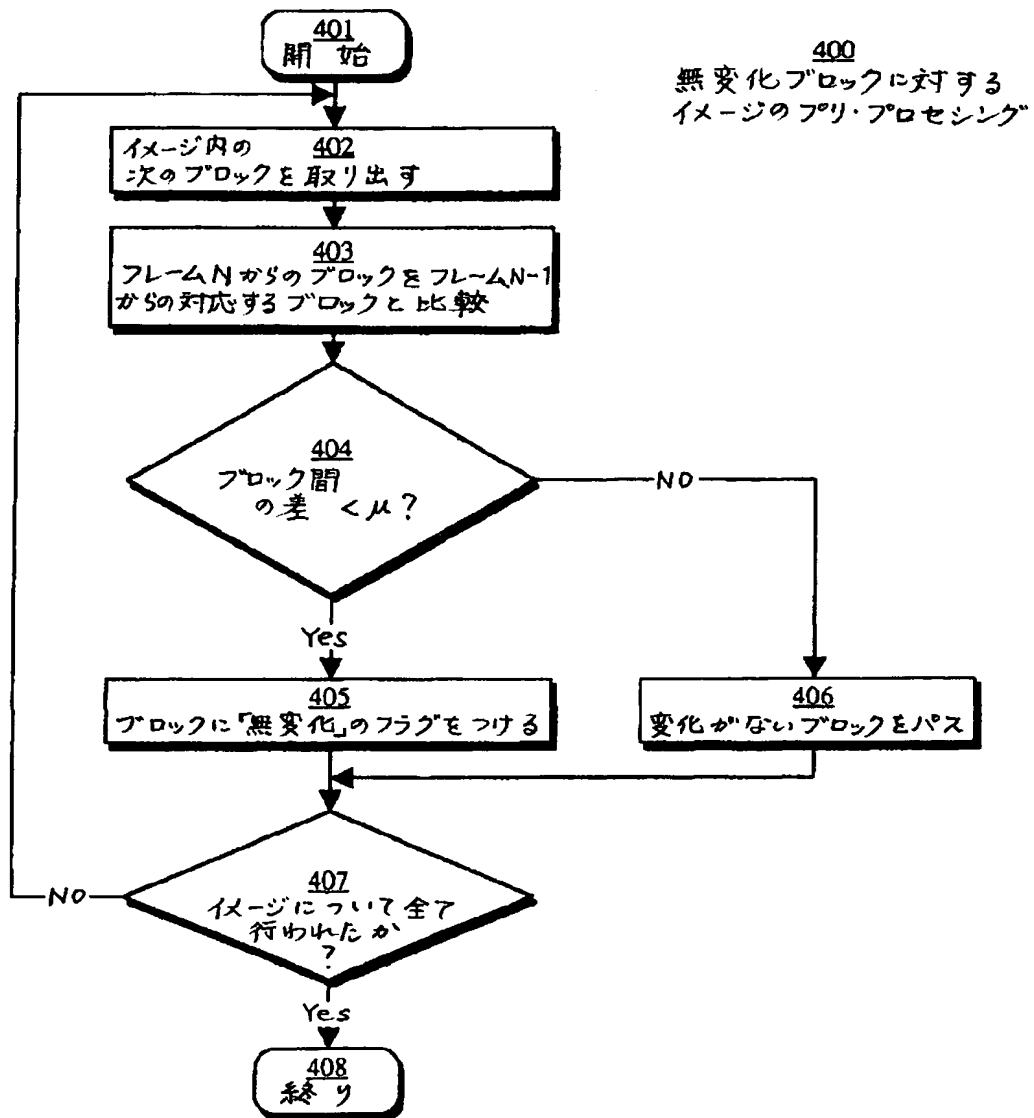
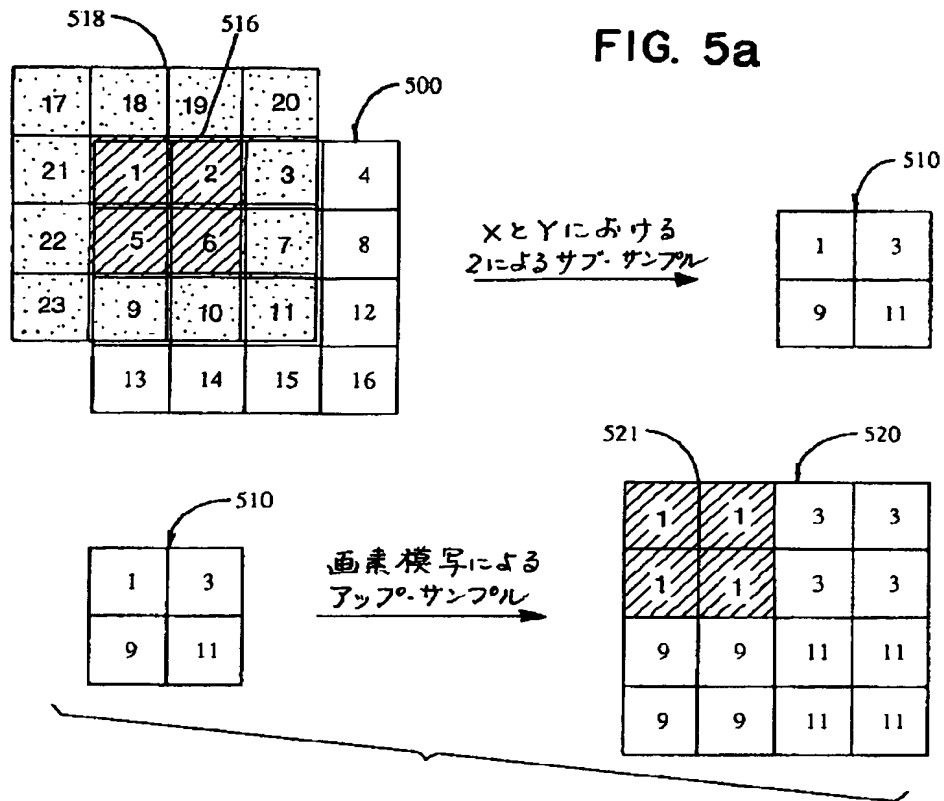
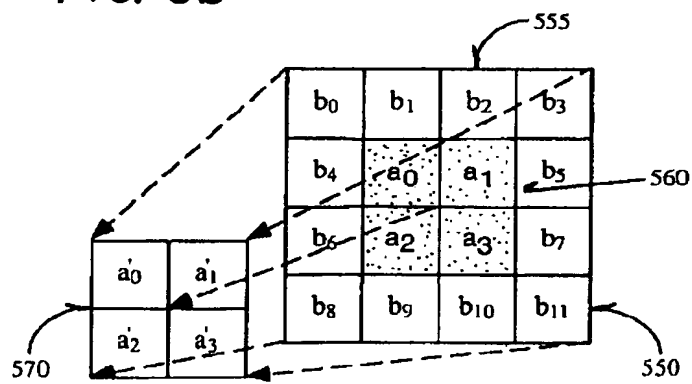


FIG. 4

【図5a】



【図5b】

FIG. 5b

【図 6】

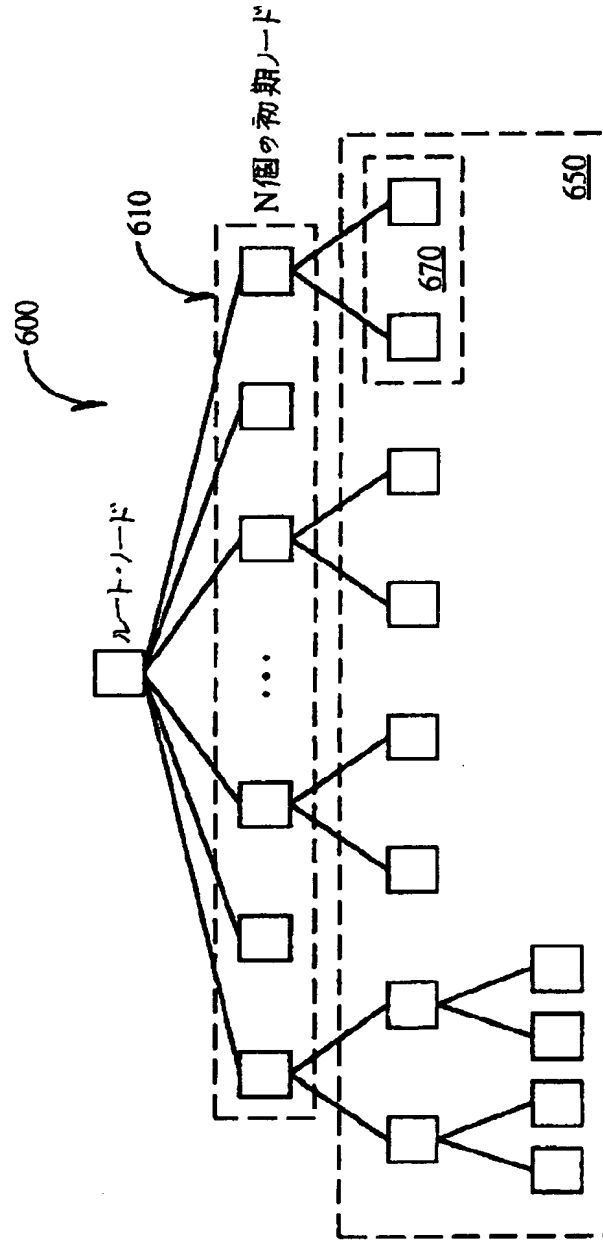


FIG. 6

【図7】

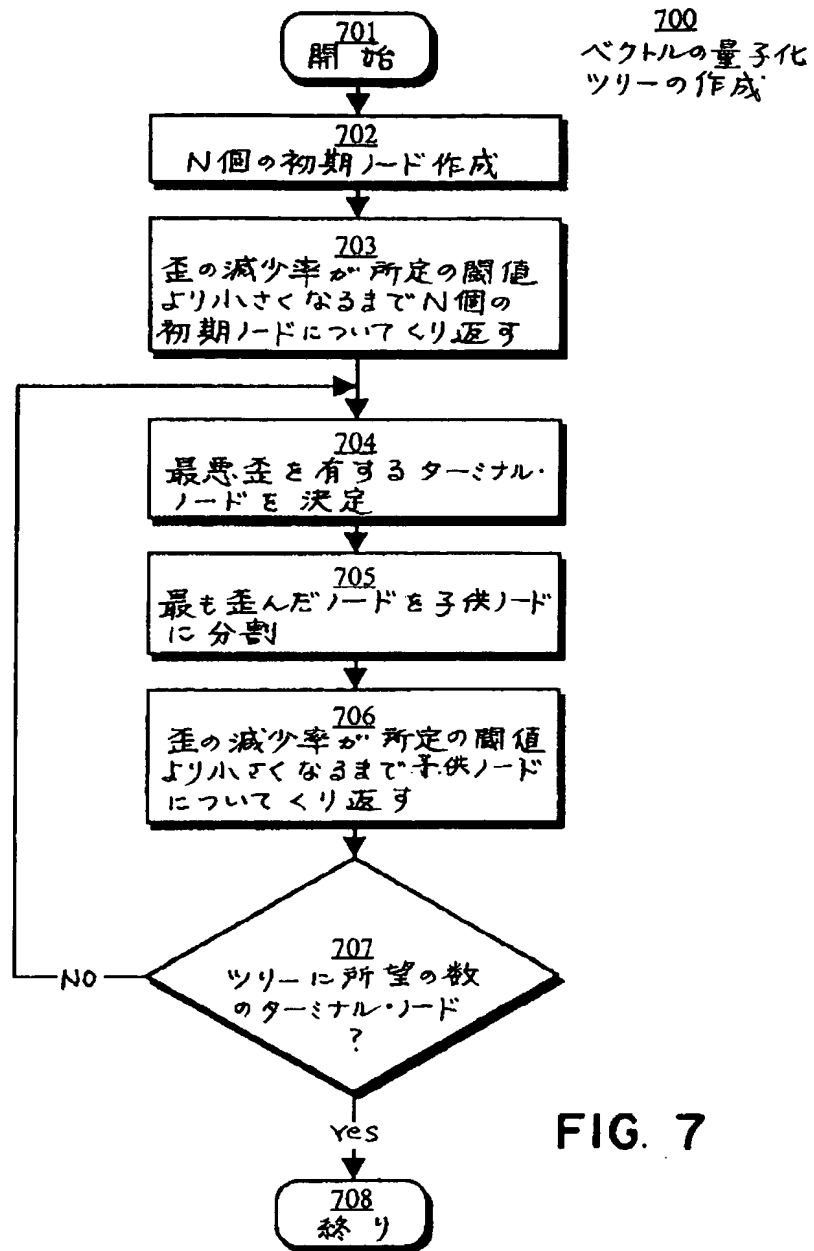


FIG. 7

【図8】

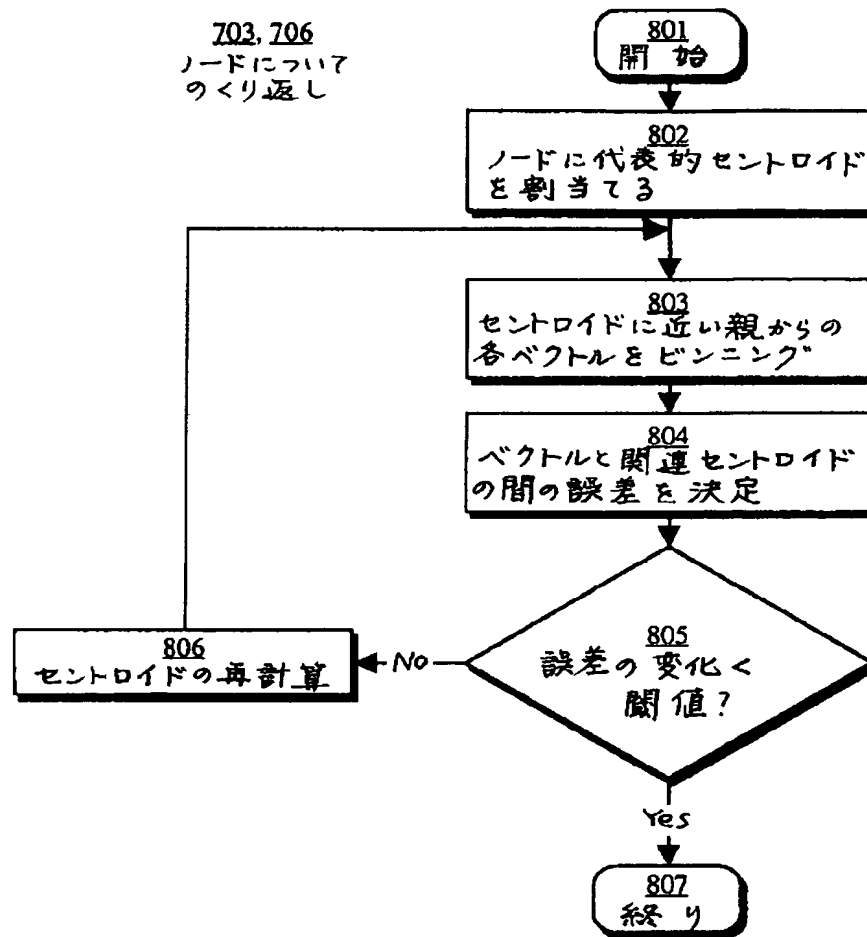


FIG. 8

【図9a】

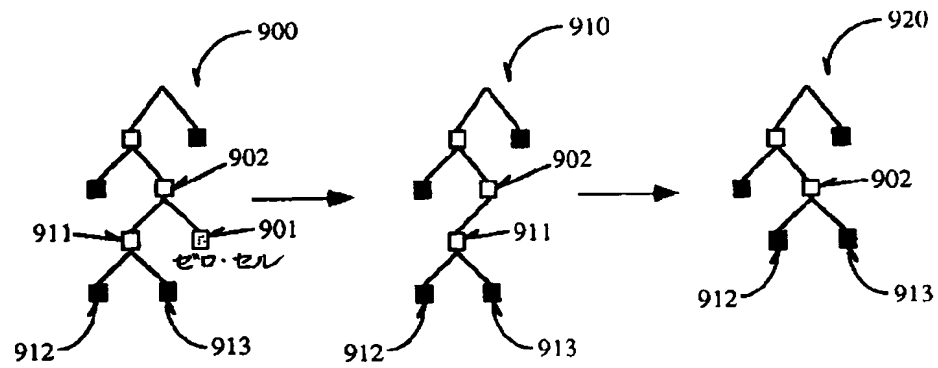


FIG. 9a

【図9b】

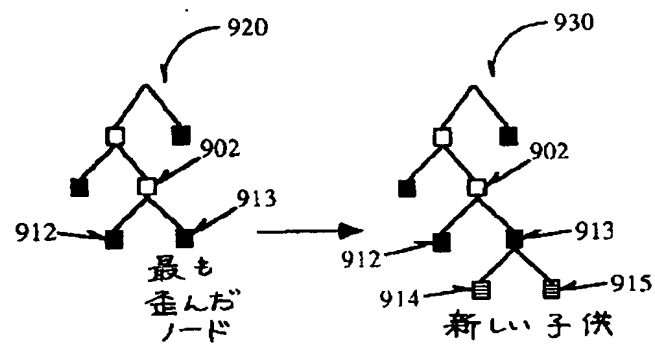


FIG. 9b

【図 10】

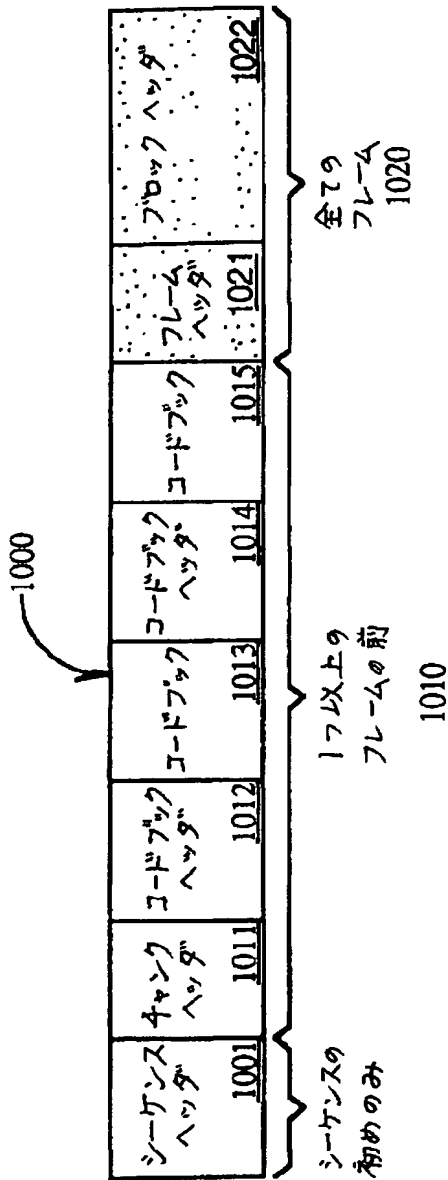


FIG. 10

【図 1 1】

シーケンス・ヘッダ

1001

シーケンス・ヘッダ ID	1101
シーケンス・ヘッダのバイト数	1102
フレーム 数	1103
予 備	1104
幅	1105
高さ	1106
バージョン	1107

FIG. 11

【図 1 2】

フレーム・ヘッダ

1021

幅	1201
高さ	1202
フレーム・タイプ	1203
サブサンプリングされたゾーン	1204

FIG. 12

【図13a】

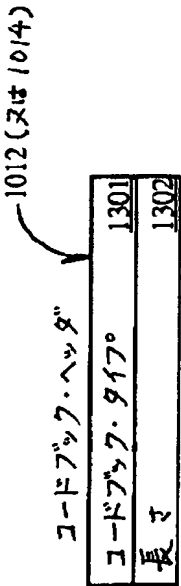


FIG. 13a

【図13b】

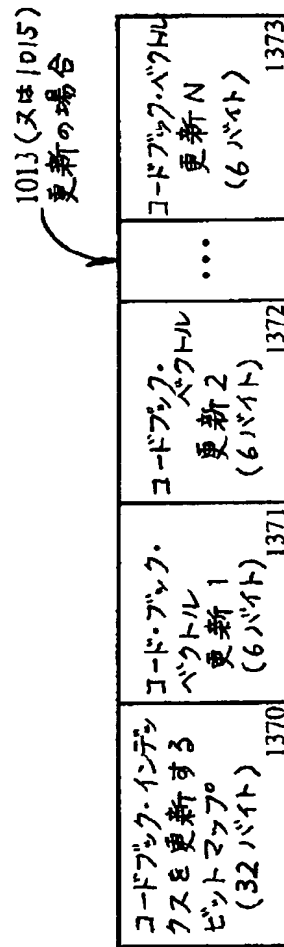


FIG. 13b

【図 1 4】

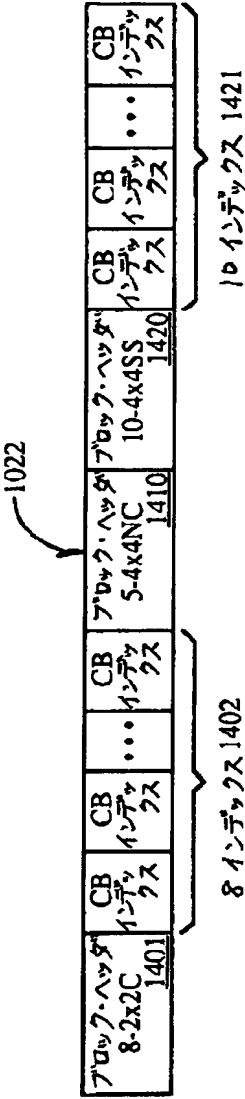


FIG. 14

【図 15】

コードブックに対するインデックスのリストを用いたインデックスのパッキング

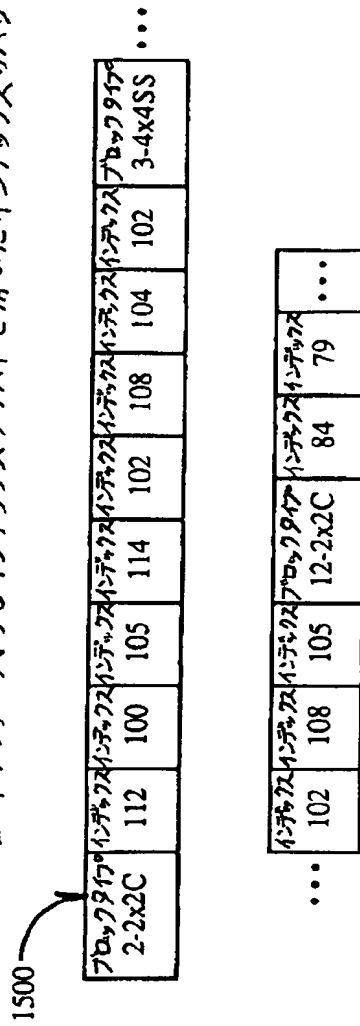


FIG. 15

【図 16】

1600

ベース・アドレスに示すオフセットにリストを用いてコードブロック・インデックスを参照する
インデックスのバッキング

ベースアドレス 107 1601	プロダクト 2-2xC 1602	オフセット +5 1603	オフセット -7 1604	オフセット -2	オフセット +7	オフセット -5	オフセット +1	オフセット -3	オフセット -5	プロダクト 3-4x4SS	...
------------------------	------------------------	---------------------	---------------------	-------------	-------------	-------------	-------------	-------------	-------------	------------------	-----

...	オフセット -5	オフセット +1	オフセット -3	ベースアドレス 82	プロダクト 12-2xC	オフセット +2	オフセット -3	...
-----	-------------	-------------	-------------	---------------	-----------------	-------------	-------------	-----

FIG. 16

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US93/08235

A. CLASSIFICATION OF SUBJECT MATTER

IPC(S) : G06K 9/36, 9/46; HO4N 7/12

US CL : 382/56; 358/133

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 382/56; 358/133

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US, A, 4,807,298 (CONTE ET AL.) 21 February 1989, col. 3, line 59-col. 6, line 54	18-77
Y		1-17
Y	US, A, 5,068,723 (DIXIT ET AL.) 26 November 1991, see entire document	1-77
Y, P	US, A, 5,194,950 (MURAKAMI ET AL.) 16 March 1993, see entire document	18-77
Y, P	us,a. 5,231,485 (ISRAELSON ET AL.) 27 July 1993, entire document	18-77

☐ Further documents are listed in the continuation of Box C.☐ See patent family annex.

* Special categories of cited documents:

A document defining the general state of the art which is not considered to be part of particular relevance

E earlier documents published on or after the international filing date

L document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

O document referring to an oral disclosure, use, exhibition or other means

P document published prior to the international filing date but later than the priority date claimed

T later documents published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

X document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

Y document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

Z document member of the same patent family

Date of the actual completion of the international search

17 December 1993

Date of mailing of the international search report

28 JAN 1994

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231

Authorized officer

CHRIS KELLEY

Facsimile No. NOT APPLICABLE

Telephone No. (703) 305-9640

フロントページの続き

(51) Int. Cl.⁶ 識別記号 庁内整理番号 F I

H 0 4 N 11/04 Z 9185-5C

(81) 指定国 EP(AT, BE, CH, DE,
DK, ES, FR, GB, GR, IE, IT, LU, M
C, NL, PT, SE), OA(BF, BJ, CF, CG
, CI, CM, GA, GN, ML, MR, NE, SN,
TD, TG), AT, AU, BB, BG, BR, BY,
CA, CH, CZ, DE, DK, ES, FI, GB, H
U, JP, KP, KR, KZ, LK, LU, MG, MN
, MW, NL, NO, NZ, PL, PT, RO, RU,
SD, SE, SK, UA, US, VN

(72) 発明者 ワング, キャサリン・シューウェイ
 アメリカ合衆国 95129 カリフォルニア
 州・サン ホゼ・ウィルミントン アヴェ
 ニュ・1072

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☐ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.